# On the Partial Semantics for Disjunctive Deductive Databases

Thomas Eiter, Nicola Leone, Domenica Saccà

Christian Doppler Labor für Expertensyteme
Technische Universität Wien
Paniglgasse 16, 1040 Wien

August 19, 1995

CD-TR 95/82

# ON THE PARTIAL SEMANTICS FOR DISJUNCTIVE DEDUCTIVE DATABASES


**T. Eiter**
Christian Doppler Lab.
Technische Universitat Wien
1040 Vienna, Austria
eiter@dbai.tuwien.ac.at

**N. Leone**
ISI-CNR
c/o DEIS-UNICAL,
87030 Rende, Italy
nik@si.deis.unical.it

**D. Saccà**
DEIS
Università della Calabria,
87030 Rende, Italy
sacca@si.deis.unical.it


preliminary report


**Abstract**

Partial stable models for deductive databases, i.e., normal function-free logic programs (also called datalog programs), have two equivalent definitions: one based on 3-valued logics and another based on the notion of unfounded set. The notion of partial stable model has been extended to disjunctive deductive databases using 3-valued logics. In this paper, a characterization of partial stable models for disjunctive datalog programs is given using a suitable extension of the notion of unfounded set.

Two interesting subclasses of partial stable models, *M-stable* (*Maximal-stable*) (also called *regular models*, *preferred extension*, and *maximal stable classes*) and *L-stable* (*Least undefined-stable*) models, are then extended from normal to disjunctive datalog programs. L-stable models are shown to be the natural relaxation of the notion total stable model; on the other hand the less strict M-stable models, endowed with a nice modularity property, may be appealing from the programming and computational point of view. M-stable and L-stable models are also compared with the *regular models* for disjunctive datalog programs recently proposed in the literature.

# 1 Introduction

Deductive databases comprise explicit data and logical rules for computing queries on the data, which amount to a logic program without function symbols, i.e., datalog programs (extended with negation) [34, 5]. The need for representing disjunctive information led to disjunctive deductive databases [23], which fertilized also work on disjunctive logic programming [22, 24]. In this paper, we address disjunctive deductive databases, which are identified with disjunctive function-free logic programs [18], and refer to them simply as programs.

To date, the *total stable* (or *2-valued stable*) model semantics [16, 26] is a dominant semantics for both normal and disjunctive programs. It is widely acknowledged that the total stable semantics is the natural meaning to be assigned to a (disjunctive) program, provided that it admits some stable models. However, a drawback of this semantics is that several meaningful programs do not admit any total stable model, which results in a loss of information. To handle these cases, Przymusinski defined the notion of *partial stable* (or *3-valued stable*) model for both normal and disjunctive programs [26]. Partial stable models exist for a wider class of programs (actually, for normal programs they always exist). Moreover, on the class of (disjunctive) stratified programs, partial stable models coincide with total stable models. For normal programs, partial stable models have been also defined using the notion of unfounded set [29, 30].

A first contribution of this paper is an equivalent definition of partial stable semantics for such programs which singles out partial stable models without the need to resort to 3-valued logics. To this end, we present a new notion of unfounded set for disjunctive programs, which generalizes the analog notion given for normal (i.e., nondisjunctive) programs in [36] (moreover, on total interpretations, it coincides with the definition given in [20] for disjunctive programs). Intuitively, as for normal programs, the notion of unfounded set can be seen as a form of CWA rule: if a set $X$ of undefined atoms is an unfounded set w.r.t. an interpretation $I$, then the truth value of the atoms in $X$ can be switched to false without violating any rule of the program.

Once defined unfounded sets, we provide an appealing characterization of 3-valued stable models in terms of unfounded sets. In particular, we show that an interpretation is a 3-valued stable model if and only if its true atoms are "justifiable" (see below) and the set of its false atoms is a maximal unfounded set. The proof of this equivalence is to be considered a relevant part of the contribution of this paper. To our knowledge, this is the first characterization of 3-valued stable models in terms of unfounded sets. It extends similar studies conducted in [36, 29, 30, 20], which investigated the relationships between unfounded sets and stable models in more restricted domains.

Despite their relevance, several objections to partial stable models came from the observation that every 3-valued model theoretic approach should satisfy two basic principles: *justifiability* (also called *foundedness*) and *minimal undefinedness* [37, 38]. Intuitively, the principle of justifiability, which is intimately related to the concept of labeling-based justification in Doyle's truth maintenance system [8], states that every positive literal which is true must be derived from the rules of the program possibly using negative literals as additional axioms. The principle of minimal undefinedness prescribes that the *undefined* truth value should be used only when necessary (i.e., a "good" semantics should tend to minimize the set of undefined atoms).

Partial stable models meet the principle of justifiability but do not satisfy the principle of minimal undefinedness. This observation led several researchers to investigate on refinements of 3-valued stable model semantics which reduce the degree of undefinedness. For normal programs, these efforts led to the definition of several notion of partial models: *M-stable models* [29, 32, 31][1], *regular models* [37, 38], *preferred extensions* [9], *maximal stable classes* [3, 39], and *L-stable models* [32, 31]. Interestingly, even if all these semantics have been independently defined, M-stable models, preferred extensions, regular models, and maximal stable classes exactly coincide, as proven in [17] and [39]. In the more general case of disjunctive programs, to our knowledge no subclass of 3-valued stable models minimizing undefinedness has been proposed. (Regular models have been extended to the disjunctive case in [38]; but, differently from the case of normal programs, a regular model is not guaranteed to be a partial stable model.)

Another contribution of this paper is the proposal of two refinements of the partial stable semantics for disjunctive programs, namely, the *M-stable* (*Maximal stable*) and the *L-stable* (*Least undefined stable*) semantics, generalizing the analogous semantics presented in [32, 31] for normal programs. Both semantics extend the partial stable semantics in the spirit of the principle of minimal undefinedness of [37, 38].

Since several researchers recognized M-stable models as an important semantics for normal programs [3, 9, 29, 32, 31, 37, 38], we start by extending M-stable models to the domain of disjunctive programs.

The M-stable models are the partial stable models which are maximal under set inclusion (where a partial model is represented by the set of ground literals true in the model). They present the following main characteristics:

1. On normal programs, M-stable models coincide with the preferred extensions of [9], the regular models of [37], the maximal stable classes of [3], and the M-stable models of [32, 31].
2. On the class of stratified (disjunctive) programs, M-stable models exactly coincide with total stable models.
3. M-stable semantics has a lower degree of undefinedness compared to 3-valued stable semantics.

Even if M-stable models reduce the degree of undefinedness of partial stable models, they do not fully agree with total stable models (i.e., even for program admitting total stable models, an M-stable model may be not total stable). A further refinement of the

[1]Note that M-stable models are called *P-stable models* in [29].

notion of M-stable model yields the L-stable models, which fully agree with total stable models when a program admit such models.

The L-stable models are the M-stable models with the minimal degree of undefinedness. They have the following peculiarities:

1. On normal programs, L-stable models coincide with the L-stable models of [32, 31].
2. On the class of stratified (disjunctive) programs, L-stable models exactly coincide with total stable models.
3. L-stable semantics has a lower degree of undefinedness compared to M-stable semantics (and also to 3-valued stable semantics).
4. On the class of programs admitting some total stable model, the L-stable models exactly coincide with the total stable models.

Thus, L-stable semantics has an even lower degree of undefinedness than M-stable semantics. Moreover, the fact that L-stable semantics may differ from total stable semantics *only* on (unstratified) programs having no total stable model, is highly desirable to approximate the total stable semantics and candidates L-stable models as the natural relaxation of total stable models.

We compare both M-stable and L-stable models with another attempt to "approximate“ total stable models for disjunctive programs due to You and Yuan [38]. In that paper, the authors extend the regular model semantics from normal to disjunctive programs. Regular models exist for every disjunctive program and are "more defined" (i.e., with a lower degree of undefinedness) than 3-valued stable models. However, a regular model is in general not a partial stable model and, even if the program admits total stable models, regular model semantics does not coincide with total stable semantics. Because of the latter drawback, regular models can not be considered a "good approximation" of total stable semantics (in case that a program has total stable models, it is widely recognized that they are the intuitive meaning of the program and every semantics should coincide on these programs).

A careful analysis of regular models reveals that a reason for their disagreement to stable models is that, differently from all the dominant semantics for datalog (and also from M-stable and L-stable semantics), they do not fully satisfy the closed world assumption. We individuate the lack of this property as a drawback of the regular model semantics and discuss the main consequences of that.

A final issue of this paper is the investigation of modularities properties of which partial semantics for disjunctive programs are endowed with. In particular, extending similar studies conducted in [34, 33, 28] for normal programs and, for disjunctive programs, in [7] and [11], we prove that both M-stable and regular models for disjunctive programs are endowed with an appealing modularity property. This modularity property allows us to enlarge the class of programs on which M-stable and regular models are known to coincide. Furthermore, the modularity property permits to derive that, for normal programs, both

M-stable and regular models can be computed bottom-up following the dependencies between the predicates of the program (in a process similar to the computation of stratified programs [34]). The latter computational property is particularly relevant in the context of deductive databases, where it is very important to reduce the amount of data to be dealt with at a time.

The main contributions of this paper can be summarized as follows.

- We present a new notion of unfounded set, which extends the classical definition of Van Gelder et al. [36] from normal to disjunctive programs.
- We provide a declarative characterization of 3-valued stable models in terms of unfounded sets. In particular, we show that an interpretation is a 3-valued stable model if and only if it is founded (i.e., all its true atoms are justifiable) and its false atoms form a maximal unfounded set.
- We extend the notions of M-stable and L-stable models from normal to disjunctive programs.
- We carry out a comparison between M-stable, L-stable and regular semantics for disjunctive programs and propose L-stable models as the natural relaxation of the notion of total stable model.
- We investigate modularity properties which hold for partial semantics. This analysis allows us to enlarge the class of programs on which M-stable and regular models are known to coincide and permits to derive useful computational properties for both M-stable and regular models.

The sequel of the paper is organized as follows. After some preliminary on disjunctive deductive databases, Section 2 presents the notion of unfounded set and provides our definition of *P-stable* (*Partial stable*) model in terms of unfounded sets. Section 3 demonstrates the equivalence between P-stable models and the 3-valued stable models of [26]. M-stable and L-stable models are defined in Section 4. In Section 5, M-stable and L-stable models are compared with the regular models for disjunctive programs [38]. In Section 6, we demonstrate some modularity properties for M-stable and regular models and apply these results to derive both semantic and computational properties of M-stable and regular semantics. Finally, Section 7 addresses some further issues and concludes the paper.

## 2 Partial Stable Models

### Disjunctive Deductive Databases

Deductive databases are described in a logical language using variables from a set $V$, predicates of nonnegative arity from a countable set $\Pi$, and constants from a finite set $\Lambda$.

An *atom* is of form $a(t_1, \ldots, t_n)$, where $a$ is a *predicate* of arity $n$ in $\Pi$ each $t_1, \ldots, t_n$ is either a constant in $\Lambda$ or a variable in $V$. A *literal* is either a *positive literal* $p$ or a *negative literal* $\neg p$, where $p$ is an atom. We use often an upper-case letter, say $L$, to denote either a positive or a negative literal. Two literals are *complementary* if they are of the form $p$ and $\neg p$, for some atom $p$. Given a literal $L$, $\neg.L$ denotes its complementary literal, and given a set $A$ of literals, $\neg.A$ denotes the set $\{\neg.L \mid L \in A\}$. A *rule* $r$ is a clause of the form

$$a_1 \vee \cdots \vee a_n \leftarrow b_1, \cdots, b_k, \neg b_{k+1}, \cdots, \neg b_m, \qquad n \geq 1,\ m \geq 0.$$

where $a_1, \cdots, a_n, b_1, \cdots, b_m$ are atoms. The disjunction $a_1 \vee \cdots \vee a_n$ is the *head* of $r$, while the conjunction $b_1, \ldots, b_k, \neg b_{k+1}, \ldots, \neg b_m$ is the *body* of $r$. We denote by $H(r)$ the set $\{a_1, \ldots, a_n\}$ of the head atoms, and by $B(r)$ the set $\{b_1, \ldots, b_k, \neg b_{k+1}, \ldots, \neg b_m\}$ of the body literals. Moreover, $B^+(r)$ and $B^-(r)$ denote the set of positive and negative literals occurring in $B(r)$, respectively. A *disjunctive deductive database* (simply *program* hereafter) is a finite set of rules. Thus, a disjunctive deductive database is a disjunctive datalog program [11]. A distinction between extensional (database) predicates and intensional (derived) predicates can be made as usual [34, 18]; for simplicity, we do not extend on this in detail.

A program is *normal* if it contains only *normal* clauses, i.e., the head contains exactly one atom ($n = 1$). A $\neg$-free (resp. $\vee$-free) program is called *positive* (resp. *normal*). An atom, a literal, a rule or program is *ground* if no variable appears in it. As common, we assume that the sets $\Pi$ and $\Lambda$ are implicitly given by the predicates and constants occurring in the considered program.

### Interpretations

We denote by $B_{\mathcal{LP}}$ the *Herbrand Base* of $\mathcal{LP}$, i.e. the set of all ground atoms constructible from the predicates appearing in the rules of $\mathcal{LP}$ and the constants occurring in $\mathcal{LP}$.

We denote by *ground*$(\mathcal{LP})$ the set of all ground instances of the rules occurring in $\mathcal{LP}$, where a *ground instance* of a rule $r$ is obtained by any substitution $\sigma$ that maps each variable $X$ in $r$ to a constant occurring in in $\mathcal{LP}$. By $Lit_{\mathcal{LP}}$ we denote the set of all literals involving atoms from *ground*$(\mathcal{LP})$, i.e. $Lit_{\mathcal{LP}} = \mathcal{B}_{\mathcal{LP}} \cup \neg.\mathcal{B}_{\mathcal{LP}}$.

An *(Herbrand) interpretation* $I$ of a program $\mathcal{LP}$ is a consistent set of ground literals, i.e. $I \subseteq Lit_{\mathcal{LP}}$ such that $I \cap \neg.I = \emptyset$. A ground literal $L$ is *true* (resp. *false*) w.r.t. $I$ if $L \in I$ (resp. $L \in \neg.I$). If a ground literal is neither true nor false w.r.t. $I$ then it is *undefined w.r.t. I*. A (possibly empty) conjunction of ground literals $C$ is *true* in $I$ if every literal in $C$ is in $I$, *false* in $I$ if there exists some literal $A$ in $C$ for which $\neg A \in I$ and *undefined* in $I$ otherwise. We denote by $I^+$ and $I^-$ the set of positive and negative literals occurring in $I$, respectively. $\overline{I}$ denotes the set of undefined atoms w.r.t. $I$ (that is, $\overline{I} = \mathcal{B}_{\mathcal{LP}} - (I^+ \cup \neg.I^-)$). The interpretation $I$ is *total* if $\overline{I}$ is empty (that is, $I^+ \cup \neg.I^- = \mathcal{B}_{\mathcal{LP}}$); otherwise, $I$ is *partial.* A *total model* of $\mathcal{LP}$ is a total interpretation

$M$ which satisfies each rule $r$ in $ground(\mathcal{LP})$, i.e., if $B(r) \subseteq M$, then $H(r) \cap M \neq \emptyset$.[2]

### Unfounded Sets

**Definition 1** *(Unfounded Set)* Let $I$ be an interpretation for $\mathcal{LP}$. A set $X \subseteq \mathcal{B}_{\mathcal{LP}}$ of ground atoms is an *unfounded set* for $\mathcal{LP}$ w.r.t. $I$ if, for each $a \in X$, at least one of the following conditions holds for each rule $r \in ground(\mathcal{LP})$ such that $a \in H(r)$:

1. $B(r) \cap (\neg.I \cup X) \neq \emptyset$, or
2. $H(r) \not\subseteq (\neg.I \cup X)$.

An unfounded set $X$ is *consistent* w.r.t. $I$ if $X \cap I^+ = \emptyset$. ∎

For normal programs unfounded sets have been introduced to draw negative inferences for the constructive definition of the well-founded model [36] (recall that the well-founded model is indeed defined as the limit of the sequence $I_0 = \emptyset$, $I_n = T_P(I_{n-1}) \cup \neg.U_P(I_{n-1})$, where $T_P$ is the immediate consequence operator and $U_P(I)$ denotes the greatest unfounded set w.r.t. $I$). Thus, they can be seen as a sort of CWA rule allowing to derive negative information (i.e., the falsity of some atoms).

Unfounded sets for disjunctive programs play essentially the same role. Indeed, if $X$ is a consistent unfounded set (i.e., $I \cup \neg.X$ is an interpretation), for every rule $r$ having an element of $X$ in the head, either the body of $r$ is false w.r.t. $I \cup \neg.X$ (Condition 1 of Definition 1), or the head of $r$ is not false w.r.t. $I \cup \neg.X$ (Condition 2 of Definition 1). Thus, if if we think in terms of classical (2-valued) interpretations, it is possible to extend $I$ by assuming that all atoms in $X$ are false,[3] without explicitly violating $r$; in fact, rule $r$ is still satisfiable: if Condition 1 applies, then $r$ is definitely satisfied (as the body is false), and if Condition 2 applies, $r$ can be satisfied by assuming that the atoms in $H(r) - X$ are true. Since the atoms in $X$ are "defined" only by such rules $r$ (under the intended assignment-based reading of rules), intuitively every atom in $X$ should be false, since the assumption that it is true is not much founded, according to a principle of closed world assumption. (For a more elaborate discussion on this, see Section 5.)

**Example 1** Consider the program $\mathcal{LP} = \{a \vee b \leftarrow\}$ and the interpretation $I = \emptyset$. The set $X = \{a\}$ is a (consistent) unfounded set of $\mathcal{LP}$ w.r.t. $I$. Indeed, the unique rule of $ground(\mathcal{LP})$ (with $a$ in the head) satisfies Condition 2 of Definition 1, as $H(r) - (X \cup \neg.I) = \{b\}$. Likewise, we can easily verify that $Y = \{b\}$ is a (consistent) unfounded set for $\mathcal{LP}$ w.r.t. $I$ (as $H(r) - (Y \cup \neg.I) = \{a\}$).
For the program $\mathcal{LP} = \{\ a \vee b \leftarrow\ ;\ a \leftarrow b\ ;\ b \leftarrow a\ \}$ and the interpretation $I = \{a, b\}$ the only unfounded set is $\emptyset$. ∎

[2]In the context of classical (2-valued) logic, where interpretations are total, (total) models are identified with their positive parts. Within the context of partial (3-valued) interpretations, however, their negative parts are mentioned as well, to stay consistent with the framework.

[3]Note that assuming false a literal which is in $I^+$ would cause inconsistency.

We remark that a first attempt for generalizing the notion of unfounded set may lead to adopt for Condition 2 the stronger requirement $(H(r)-X)\cap I^{+}\neq\emptyset$, i.e., an atom from the head of $r$ apart from $X$ is true in $I$. This definition has been adopted in [20], where it unfounded set are used to characterize 2-valued stable models. However, this (stronger) condition turns out to be too strong to capture unfoundedness appropriately with respect to 3-valued semantics of rules that we consider later. Namely, if the head of a rule as well as its body is undefined in $I$, i.e., neither definitely true nor false, we nevertheless can conclude that –in the extremal case– all atoms in the head except one can be false, without definitely violating the rule (which happens if the body is true and the head is false). Arriving at such a conclusion is essential, however, in a form of the closed world assumption which is inherent to 3-valued stable semantics.

It is worth pointing out that the above definition generalizes the standard definition of unfounded set given for normal programs in [36]: $X$ is a *(standard) unfounded set* for a normal program $\mathcal{LP}$ w.r.t. $I$ if, for each $a\in X$, for each rule with head $a$, $B(r)\cap(\neg.I\cup X)\neq\emptyset$ holds.

**Proposition 1** *Let $\mathcal{LP}$ be a normal program and $I$ an interpretation for $\mathcal{LP}$. $X\subseteq\mathcal{B}_{\mathcal{LP}}$ is an unfounded set for $\mathcal{LP}$ w.r.t. $I$ according with Definition 1 if and only if it is a standard unfounded set for $\mathcal{LP}$ w.r.t. $I$.*

**Proof.** For a normal program, Condition 2 of Definition 1 is never satisfied since $H(r)\subseteq X$, as $H(r)$ is made of a single element which belongs to $X$. Thus, for a normal program, Definition 1 reduces to the standard definition of unfounded set. ∎

It is known that for normal programs, the union of two unfounded sets is an unfounded set as well. This property does not extend to disjunctive programs.

**Example 2** Consider again the program $\mathcal{LP}=\{a\vee b\leftarrow\}$ and the interpretation $I=\emptyset$. In Example 1 we have seen that both $\{a\}$ and $\{b\}$ are unfounded sets for $\mathcal{LP}$ w.r.t. $I$, but we can easily verify that the union $X=\{a,b\}$ is not. Indeed, take any element in $X$, say $a$; although $a$ is not false w.r.t. $I$, $b$ is not unfounded as $a$ is also in $X$ (so that Condition 2 of Definition 1 does not hold). Intuitively, this represents the fact that we can falsify either $a$ or $b$, but at least one of them must remain not false (otherwise the rule $a\vee b\leftarrow$ becomes unsatisfiable). ∎

Since the union of all unfounded sets is not necessarily an unfounded set, we cannot define the greatest unfounded set of an interpretation; instead, we next provide a notion of maximal unfounded set.

A consistent unfounded set $X$ w.r.t. $I$ is a *maximal unfounded set* for $\mathcal{LP}$ w.r.t. $I$ if no proper superset $Y$ of $X$ is a consistent unfounded set for $\mathcal{LP}$ w.r.t. $I$.

**Example 3** Let $\mathcal{LP}=\{a\vee b\leftarrow\}$. Both $X_1=\{a\}$ and $X_2=\{b\}$ are maximal unfounded sets for $\mathcal{LP}$ w.r.t. $I=\emptyset$.

$X_1$ is the only maximal unfounded set for $\mathcal{LP}$ w.r.t. the interpretation $\{b\}$ (note that $X_2$ is not a consistent unfounded set w.r.t. $\{b\}$).

Let $\mathcal{LP}_1$ be the following program:

$$\begin{array}{rl} a \vee b & \leftarrow \\ d & \leftarrow \neg d, \neg b \\ a \vee c & \leftarrow \neg d \end{array}$$

$M_1 = \{a,\ \neg b,\ \neg c\}$ has the single maximal unfounded set $X_1 = \{b, c\}$; $M_2 = \{b,\ c,\ \neg a,\ \neg d\}$ has the single maximal unfounded $X_2 = \{a,\ d\}$. ∎

Intuitively, a maximal unfounded set $X$ is a maximal set of atoms that can be assumed false without violating the satisfiability of the rules of the program that contain some atom from $X$. Consistency of the unfounded set $X$ ensures that the assumption of the falsity of the literals in $X$ is consistent with $I$ (that is, $I \cup \neg.X$ is an interpretation).

For instance, we have seen in the above examples that for the program $\mathcal{LP} = \{a \vee b \leftarrow\}$ both $\{a\}$ and $\{b\}$ are (consistent) unfounded sets w.r.t. $I = \emptyset$ while $\{a, b\}$ is not an unfounded set. Indeed, we can assume the falsity of either $a$ or $b$ without violating the satisfiability of $a \vee b \leftarrow$ (which can be satisfied by making true $b$ or $a$, respectively); but we cannot assume the falsity of both $a$ and $b$ at the same time, as in the latter case we would have no way to satisfy rule $a \vee b \leftarrow$.

## P-Stable Models

Once we provided a notion of unfounded set for disjunctive programs, we define P-stable models for such programs by extending the respective definition given in [32, 31] for normal programs.

To proceed, we need the concept of foundedness of an interpretation. *Foundedness*, called *justifiability* in [37, 38], basically prescribes that every positive literal in an interpretation must be be derived from the rules possibly using negative literals as additional axioms.

Let for any positive (i.e., $\neg$-free) program $P$ denote $\mathrm{MM}(P)$ the set of the minimal total models of $P$ [23], where a total model $M$ is *minimal* iff there does not exist a total model $N$ of $P$ such that $N^+ \subset M^+$. Moreover, let for any program $\mathcal{LP}$ and interpretation $I$ be $\mathcal{LP}(I)$ the program obtained from $ground(\mathcal{LP})$ as follows: (i) remove all rules $r$ having a negative literal $\neg p \in B(r)$ such that $\neg p \notin I^-$, and (ii) by remove all negative literals from the remaining rules.

**Definition 2** *An interpretation $I$ of a program $\mathcal{LP}$ is* founded *iff $I^+ = M^+$ for some $M \in \mathrm{MM}(\mathcal{LP}(I))$.*

Notice that $\mathcal{LP}(I)$ is essentially the 2-valued Gelfond-Lifschitz transform [16], carried out in the context of partial interpretations.

**Example 4** Consider the program $\mathcal{LP}_1$ of Example 3. Interpretation $M_1 = \{a,\ \neg b,\ \neg c\}$ is founded. Indeed, $\mathcal{LP}_1(M_1)$ is the following program:

$$a \vee b \ \leftarrow$$

and there exists a minimal total model $N$ of $\mathcal{LP}_1(M_1)$ such that $N^+ = M_1^+$, namely $N = \{a,\ \neg b\}$. For $M_2 = \{b,\ c,\ \neg a,\ \neg d\}$, $\mathcal{LP}_1(M_2)$ is as follows:

$$\begin{array}{l} a \vee b \ \leftarrow \\ a \vee c \ \leftarrow \end{array}$$

Also $M_2$ is founded, since $N = \{b, c, \neg a\}$ is a minimal total model of $\mathcal{LP}_1(M_2)$ and $N^+ = M_2^+$. ∎

For total interpretations foundedness alone is sufficient to characterize (2-valued) stable models, which are indeed defined as the total founded interpretations of the program [16, 26]. However, as for normal programs [29], foundedness alone is not sufficient to single out 3-valued stable models, as some conditions on the false literals are needed. Intuitively, what must be imposed on the false literals for a founded interpretation $I$ to be a (partial) stable model is the consistency with the closed world assumption. In other words, every false literal in $I$ must be definitely not derivable from the rules of the program (assuming $I$), and no further literal must be declarable false without violating some rule of the program. As we informally observed above, this consistency with the CWA is precisely formalized by the notion of maximal unfounded set.

**Definition 3** *(P-stable Model)* Let $\mathcal{LP}$ be a program and $M$ an interpretation of it. Then $M$ is a *P-stable* (*partial stable*) model of $\mathcal{LP}$ if the following conditions hold:

$(a)$ $M$ is founded,

$(b)$ $\neg.M^-$ is a maximal unfounded set for $\mathcal{LP}$ w.r.t. $M$. ∎

Note that Condition (a) characterizes the true atoms of $M$, imposing that they have to be "justified" [37], while condition (b) characterizes the false atoms of $M$, requiring that they must be all and only the atoms that can be assumed false consistently with $M$ and with the rules of the program (further negative assumptions would make some rule unsatisfiable).

**Example 5** Consider the program $\mathcal{LP}_1$ of Example 3. Both $M_1 = \{a,\ \neg b, \neg c\}$ and $M_2 = \{b,\ c,\ \neg a,\ \neg d\}$ are P-stable models of $\mathcal{LP}_1$. Indeed, we have seen in Example 4 that $M_1$ and $M_2$ are founded, and Example 3 showed that $\neg.M_1^-$ and $\neg.M_2^-$ are maximal unfounded sets w.r.t. $M_1$ and $M_2$, respectively. On the other hand, the interpretation $M_3 = \{a, \neg d\}$ is not a P-stable model of $\mathcal{LP}_1$. While $M_3$ is founded (note that $\mathcal{LP}_1(M_3) = \{a \vee b \leftarrow\ ,\ a \vee c \leftarrow\ \}$), $\neg.M_3^-$ is not a maximal unfounded set w.r.t. $M_3$; indeed, $M_3$ has the single maximal unfounded set $X = \{b, c\}$. ∎

We close this section justifying that Definition 3 coincides on normal programs with the definition given in [32].

**Proposition 2** *Let $\mathcal{LP}$ be a normal program. $M$ is a P-stable model for $\mathcal{LP}$ according with Definition 3 if and only if $M$ is a P-stable model according with [32].*

**Proof.** In [32] an interpretation $M$ is a P-stable model if it is founded (according to the definition there) and $\neg.M^- = GUS_{\mathcal{LP}}(M)$, i.e., the greatest unfounded set of $\mathcal{LP}$ w.r.t. $M$ (which always exists for a normal program). Our notion of foundedness amounts on normal programs to the notion in [32]. Moreover, for normal programs it is easily checked that any unfounded set $X$ w.r.t. a founded interpretation $I$ satisfies $X \cap I = \emptyset$, and hence each unfounded set $X$ w.r.t. $I$ is consistent. Therefore, if $M$ is founded, $\neg.M^- = GUS_{\mathcal{LP}}(M)$ if and only if $\neg.M^-$ is a maximal unfounded set for $\mathcal{LP}$ w.r.t. $M$. Consequently, the two definitions coincide. ∎

As a corollary we obtain the following result.

**Corollary 1** *Let $\mathcal{LP}$ be a normal program. $M$ is a P-stable model for $\mathcal{LP}$ if and only if it is a 3-valued stable model for $\mathcal{LP}$ according with [26].*

**Proof.** In [30] it is proven that on normal programs the P-stable models of [32] coincide with the 3-valued stable models of [26]. Thus, the result follows from Proposition 2. ∎

Corollary 1 shows that P-stable models coincide with 3-valued stable models on normal programs. The next section proves that this equivalence result extends to disjunctive programs.

# 3 P-stable and 3-Valued Stable Models

In this section we show that the definition of P-stable model (Definition 3) is equivalent to the definition of 3-valued stable model as given in [26]. To this end, following [26], we define a 3-valued logic with $T$ *(True)*, $F$ *(False)*, and $U$ *(Undefined)*, ordered as $F < U < T$. Given a program $\mathcal{LP}$, an interpretation $I$ of $\mathcal{LP}$, and a ground literal $L$,

$$value_I(L) = \begin{cases} T, & \text{if } L \in I; \\ F, & \text{if } L \in \neg.I; \\ U, & \text{otherwise (i.e., } L \in \overline{I} \cup \neg.\overline{I}). \end{cases}$$

The value of a conjunction $C = L_1, \ldots, L_n$ of ground literals is the minimal value of the literals in the conjunction (i.e., $value_I(C) = min(\{value_I(L_i) \mid 1 \leq i \leq n\})$ ); if $C$ is empty, then we assume $value_I(C) = T$. The value of a disjunction $D = L_1 \vee \cdots \vee L_n$ of ground literals is the maximum value of the literals in the disjunction (i.e., $value_I(D) = max(\{value_I(L_i) \mid 1 \leq i \leq n\})$ ). A ground rule $r$ is *satisfied* by $I$ if $value_I(H(r)) \geq value_I(B(r))$.

**Definition 4** *(3-valued Model)[26]* Let $\mathcal{LP}$ be a program and $M$ be an interpretation for $\mathcal{LP}$. Then, $M$ is a 3-valued model for $\mathcal{LP}$ if every rule $r$ in $ground(\mathcal{LP})$ is satisfied by $M$. ∎

The definition of 3-valued stable model involves a 3-valued version of the Gelfond-Lifschitz (GL) transformation [26] and resorts to the notion of 3-valued program that we report next. A *3-valued program* is a program $\mathcal{LP}$ where the constants $T$, $U$ and $F$ may occur as body literals in the rules of $\mathcal{LP}$. We assume that in every interpretation $I$ the literal $T$ is true, the literal $F$ is false, and the literal $U$ is undefined. Thus, such literals can be thought of as already interpreted ground literals.

Given an interpretation $I$ for a program $\mathcal{LP}$, the *GL-transformation* $\frac{\mathcal{LP}}{I}$ of $\mathcal{LP}$ w.r.t. $I$ is the positive 3-valued program obtained from $ground(\mathcal{LP})$ by replacing in the body of every rule all negative literals which are true (resp. undefined, false) by $T$ (resp. $U$, $F$). An interpretation $M$ for $\mathcal{LP}$ is a *3-valued model for* $\frac{\mathcal{LP}}{I}$ if it satisfies every rule in $\frac{\mathcal{LP}}{I}$. A 3-valued model $M$ for $\frac{\mathcal{LP}}{I}$ (resp. $\mathcal{LP}$) is *minimal* if there does not exist a 3-valued model $M_1 \neq M$ for $\frac{\mathcal{LP}}{I}$ (resp. $\mathcal{LP}$) such that $M_1^+ \subseteq M^+$ and $M_1^- \supseteq M^-$. In [26] it has been proven that $\frac{\mathcal{LP}}{I}$ has at least one minimal 3-valued model.

**Definition 5** *(3-Valued Stable Model)[26]* Let $\mathcal{LP}$ be a program and $M$ be a 3-valued model for $\mathcal{LP}$. Then, $M$ is a *3-valued stable model* for $\mathcal{LP}$ iff $M$ is a minimal 3-valued model of $\frac{\mathcal{LP}}{M}$. ∎

Notice that each 3-valued stable model of $\mathcal{LP}$ is a 3-valued model of $\mathcal{LP}$, and in fact a minimal 3-valued model [26].

Even if Definitions 5 and 3 are apparently quite different (the former relies on three valued logic, while the latter is based on unfounded sets), they yield exactly the same notion of stable model.

**Example 6** Consider the program $\mathcal{LP}_1$ of Example 3:

$$\begin{array}{rl} a \vee b & \leftarrow \\ c & \leftarrow a,\ d \\ d & \leftarrow \neg c \end{array}$$

We have seen in Example 5 that $M_1 = \{a,\ \neg b,\ \neg c\}$ is a P-stable model for $\mathcal{LP}_1$. It is easy to see that $M_1$ is also a 3-valued stable model for $\mathcal{LP}_1$. Indeed, $\frac{\mathcal{LP}_1}{M_1}$ is the following 3-valued program:

$$\begin{array}{rl} a \vee b & \leftarrow \\ d & \leftarrow U,\ T \\ a \vee c & \leftarrow U, \end{array}$$

and $M_1$ is a minimal 3-valued model of $\frac{\mathcal{LP}}{M_1}$. Similarly, $M_2 = \{b,\ c,\ \neg a,\ \neg d\}$, which has been proven to be a P-stable model of $\mathcal{LP}_1$ in Example 5, is a 3-valued stable model for

$\mathcal{LP}_1$, as it is a minimal 3-valued model of $\frac{\mathcal{LP}_1}{M_2}$, which is as follows:

$$\begin{array}{rl} a \vee b & \leftarrow \\ d & \leftarrow T,\ F \\ a \vee c & \leftarrow T. \end{array}$$

However, the interpretation $M_3 = \{a, \neg d\}$ is not a 3-valued stable model of $\mathcal{LP}_1$; in fact, $M_3$ is not a 3-valued model of $\mathcal{LP}_1$ (it violates the rule $d \leftarrow \neg d, \neg b$), and hence it can not be a 3-valued stable model of $\mathcal{LP}_1$. ∎

The sequel of the section is devoted to formally demonstrate the equivalence between Definitions 5 and 3. The proof of this equivalence requires some preliminary result.

**Lemma 1** *Let $M$ be a P-stable model for a program $\mathcal{LP}$. Then, $M$ is a 3-valued model for both $\mathcal{LP}$ and $\frac{\mathcal{LP}}{M}$.*

**Proof.** It is immediately recognized that $M$ is a 3-valued model for $\mathcal{LP}$ if and only if it is a 3-valued model for $\frac{\mathcal{LP}}{M}$. It is thus sufficient to prove that $M$ is a 3-valued model for $\mathcal{LP}$. Let $r$ be a rule in $ground(\mathcal{LP})$. We have to prove that: (a) $value_M(H(r)) = F \Rightarrow value_M(B(r)) = F$, and (b) $value_M(H(r)) = U \Rightarrow value_M(B(r)) \neq T$.
(a). If $value_M(H(r)) = F$ then all atoms in the head of $r$ are false. Since $M$ is a P-stable model, from Definition 3, $\neg.M^-$ is an unfounded set w.r.t. $M$. Therefore, $r$ (whose head contains elements in $\neg.M^-$) must satisfy at least one of the unfoundedness conditions of Definition 1 for $X = \neg.M^-$ w.r.t. $M$. Now, Condition 2 cannot be verified, as $H(r) \subseteq \neg.M^-$. Therefore, $B(r) \cap \neg.M \neq \emptyset$, and, as a consequence, $value_M(B(r)) = F$ (note that, since $X = \neg.M^-$, $\neg.M \cup X = \neg.M$).
(b). We prove the contrapositive: $value_M(B(r)) = T \Rightarrow value_M(H(r)) \neq U$. Let $value_M(B(r)) = T$ hold. Then, the rule $r'$ obtained from $r$ by deleting all negative body literals is in $\mathcal{LP}(M)$, as all negative body literals of $r$ are true w.r.t. $M$. The body of $r'$ is true w.r.t. $M$ (as $B(r') \subseteq B(r) \subseteq M$). Since $M$ is founded, by Definition 3 $M^+ = N^+$ for some $N \in \mathrm{MM}(\mathcal{LP}(M))$; consequently, $B(r') \subseteq M$ implies that $H(r') \cap M \neq \emptyset$. Therefore, $value_M(H(r)) = T$ (hence, $value_M(H(r)) \neq U$), as $H(r') = H(r)$. ∎

We are now in the position to prove that every P-stable model is a 3-valued model.

**Theorem 1** *Let $M$ be a P-stable model for a program $\mathcal{LP}$. Then, $M$ is a 3-valued stable model for $\mathcal{LP}$.*

**Proof.** From Definition 5, we have to demonstrate that $M$ is a minimal 3-valued model for $\frac{\mathcal{LP}}{M}$. By Lemma 1, it remains to prove only the minimality of $M$. We proceed by contradiction. Assume that $M_1 \neq M$ is a 3-valued model for $\frac{\mathcal{LP}}{M}$ such that $M_1^+ \subseteq M^+$ and $M_1^- \supseteq M^-$. We proceed as follows: we first prove that (a) $M_1^+$ cannot be a proper subset of $M^+$ and after that, using (a), we show that (b) $M_1^-$ cannot be a proper superset of $M^-$.

(a). Assume that $M_1^+ \subset M^+$ and let $N$ be the total interpretation of $\mathcal{LP}(M)$ such that $N^+ = M_1^+$ (and $N^- = \mathcal{B}_{\mathcal{LP}} - M_1^+$). We demonstrate that $N$ is a total model for $\mathcal{LP}(M)$, which contradicts the foundedness of $M$ given for $M$ by Definition 3. Let $r$ be any rule in $\mathcal{LP}(M)$, let $r'$ be a rule in $ground(\mathcal{LP})$ from which $r$ comes (i.e., $r'$ is the rule $H(r) \leftarrow B^+(r)$), and $r''$ be the GL-transformed of $r'$ in $\frac{\mathcal{LP}}{M}$. Observe first that, since $r$ is in $\mathcal{LP}(M)$, all negative literals in $B(r')$ are true w.r.t. $M$ and, as a consequence, they have been replaced in $r''$ by $T$ (and they do not appear in $r$). Therefore, the body of $r$ is true w.r.t. $N$ (i.e., $B(r) \subseteq N$) iff the body of $r''$ is true in $M_1$ (i.e., $value_{M_1}(r'') = T$).

Now, if $B(r) \not\subseteq N$ then $r$ is satisfied by $N$, as its body is false. On the other hand, if $B(r) \subseteq N$, then $B(r'')$ is true w.r.t. $M_1$ (i.e., $value_{M_1}(r'') = T$); since $M_1$ is a 3-valued model for $\frac{\mathcal{LP}}{M}$, $H(r'')$ is true w.r.t. $M_1$, i.e., $H(r'') \cap M_1^+ \neq \emptyset$. As $H(r) = H(r'')$, it follows that $r$ is satisfied by $N$. Hence, $N$ is a total model for $\mathcal{LP}(M)$ such that $N^+ \subseteq M^+$. This raises a contradiction to the fact that $M$ is founded.

(b). From point (a) above, we know that $M_1^+ = M^+$. We next prove that $\neg.M_1^-$ is an unfounded set w.r.t. $M$. Thus, if $M_1^-$ is a proper superset of $M^-$, then Condition (b) of Definition 3, i.e. the maximality of the unfounded set $M^-$, is contradicted.

To demonstrate that $\neg.M_1^-$ is an unfounded set w.r.t. $M$, we have to prove that, given any rule $r$ in $ground(\mathcal{LP})$ having an atom $a$ from $\neg.M_1^-$ in the head, $r$ satisfies at least one of the conditions of Definition 1 for $X = \neg.M_1^-$ w.r.t. $M$. Let $r'$ be the GL-transformed of $r$ in $\frac{\mathcal{LP}}{M}$. If the body of $r'$ is false w.r.t. $M_1$, then $r$ satisfies Condition 1 of Definition 1 for $X$ w.r.t. $M$. Otherwise (i.e., the body of $r'$ is not false w.r.t. $M_1$), since $M_1$ is a 3-valued model for $\frac{\mathcal{LP}}{M}$, the head of $r'$ contains some literal which is not false w.r.t. $M_1$ (that is, $H(r') - \neg.M_1^- \neq \emptyset$). Therefore, as $H(r') = H(r)$ and $X \cup \neg.M^- = \neg.M_1^-$, $r$ satisfies Condition 2 of Definition 1 for $X$ w.r.t. $M$. Thus, $\neg.M_1^-$ is an unfounded set w.r.t. $M$. Consequently, $M_1^-$ cannot be a proper superset of $M^-$. (We are done.) ∎

The proof of the converse of Theorem 1 is divided in the following two lemmata.

**Lemma 2** *Every 3-valued stable model is founded.*

**Proof.** Let $M$ be a 3-valued stable model for $\mathcal{LP}$. By contradiction, assume that the total interpretation $N$ of $\mathcal{LP}(M)$ such that $N^+ = M^+$ is not a minimal total model of $\mathcal{LP}(M)$. Then, there exists a total model $N1$ for $\mathcal{LP}(M)$ such that $N1^+$ is a proper subset of $M^+$. We show that the interpretation $J = N1^+ \cup M^-$ is a 3-valued model for $\frac{\mathcal{LP}}{M}$; this implies that $M$ is not a minimal 3-valued model of $\frac{\mathcal{LP}}{M}$, and hence contradicts that $M$ is a 3-valued stable model of $\mathcal{LP}$.

Let $r$ be any rule in $\frac{\mathcal{LP}}{M}$ and $r'$ be a rule in $ground(\mathcal{LP})$ from which $r$ is obtained. We show by considering the three possible values of $B(r)$ in $J$ that $r$ is satisfied in $J$, which means that $J$ is a 3-valued model of $\frac{\mathcal{LP}}{M}$, and raises the desired contradiction: (a) $B(r)$ is false in $J$: Then, $r$ is clearly satisfied by $J$. (b) $B(r)$ is true in $J$: Then, all negative literals of the body of $r'$ must be true in $M$. Thus, a rule $r''$ in $\mathcal{LP}(M)$ exists such that $B(r'') = B^+(r')$ and $H(r'') = H(r')$. $B(r'')$ is clearly true w.r.t. $N1$; hence, $H(r'')$ is true in $N1$ (as $N1$ is a model of $\mathcal{LP}(M)$), and, since $H(r) = H(r'')$ and $J^+ = N1^+$, $H(r)$ is true w.r.t. $J$. Thus, $r$ is satisfied in $J$. (c) $B(r)$ is undefined w.r.t. $J$: Assume that $H(r)$ is false w.r.t. $J$. Then, $H(r) \subseteq \neg.M^-$. Therefore, $H(r')$ is false w.r.t. $M$. Since $M$ is a 3-valued stable model for $\mathcal{LP}$, it is a 3-valued model of $\mathcal{LP}$ and hence $B(r')$ is false w.r.t. $M$. Thus, $B(r)$ is false w.r.t. $J$, as either it contains the constant $F$ (if a negative literal of $r$ is false w.r.t. $M$) or it contains a positive literal which is in $M^-$ (which coincides with $J^-$). Hence, by (a)–(c), $r$ is satisfied in $J$. (We are done.) ∎

**Lemma 3** *If $M$ is a 3-valued stable model, then $\neg.M^-$ is a maximal unfounded set w.r.t. $M$.*

**Proof.** Let $M$ be a 3-valued stable model for $\mathcal{LP}$. It is easy to see that $\neg.M^-$ is an unfounded set for $\mathcal{LP}$ w.r.t. $M$. Indeed, since $M$ is a 3-valued model for $\mathcal{LP}$, every rule with a literal from $\neg.M^-$ in the head either has a false body (if all head literals are false) or its head contains some literal which is not in $\neg.M^-$. To see that $\neg.M^-$ is a maximal unfounded set, i.e., there does not exist an $X$ with $\emptyset \neq X \subseteq \overline{M}$ such that $X \cup \neg.M^-$ is an unfounded set w.r.t. $M$, we demonstrate that the existence of such an $X$ entails $J = \neg.X \cup M$ is a 3-valued model for $\frac{\mathcal{LP}}{M}$, which contradicts the minimality of $M$ for $\frac{\mathcal{LP}}{M}$ and hence that fact that $M$ is a 3-valued stable model of $\mathcal{LP}$. To this end, consider any rule $r$ in $\frac{\mathcal{LP}}{M}$, which is obtained from a rule $r'$ in $ground(\mathcal{LP})$. We show that $r$ is satisfied in $J$. As a consequence, $J$ is a 3-valued of $\frac{\mathcal{LP}}{M}$, which raises the desired contradiction.

Clearly, $value_J(B(r)) \leq value_M(B(r))$. To show that $r$ is satisfied in $J$, it thus suffices to show that $value_J(H(r)) < value_M(H(r))$ implies $value_J(B(r)) = F$. Under the hypothesis, $H(r) \cap X \neq \emptyset$. Since $J^+ = M^+$, it follows that $value_J(H(r)) = F$. Since $X \cup \neg.M^-$ is an unfounded set w.r.t. $M$ and $H(r) = H(r')$, it follows by Condition 1 of Definition 1 that $B(r') \cap (\neg.M \cup X \cup \neg.M^-) \neq \emptyset$, i.e. $B(r') \cap \neg J \neq \emptyset$ (note that $J = M \cup \neg.X \cup M^-$). Consequently, $B(r')$ is false in $J$, which clearly means that $value_J(B(r)) = F$. Thus, $r$ is satisfied in $J$. (We are done.) ∎

We can finally state the main theorem of this section.

**Theorem 2** *$M$ is a P-stable model for a program $\mathcal{LP}$ if and only if $M$ is a 3-valued stable model for $\mathcal{LP}$.*

**Proof.** Follows from Theorem 1 and Lemmata 2 and 3. ∎

Thus, Definition 3 can be seen as a characterization of 3-valued stable models which does not resort to 3-valued logic.

# 4 Partial Stable Models which Minimize Undefinedness

A major objection against P-stable models is the fact that they leave more atoms undefined than necessary [37, 38].

The simplest way to select the P-stable models which are closer to the principle of minimal undefinedness is to discard those properly included in some other P-stable.

**Definition 6** *(M-stable Model)* A P-stable model $M$ is *M-stable* (*Maximal stable*) if there exists no P-stable model $N$ of $\mathcal{LP}$ which is a proper superset of $M$. ∎

**Example 7** Consider the following program $\mathcal{LP}$.

$$\begin{aligned} a &\leftarrow \neg b \\ b &\leftarrow \neg a \\ c \vee d &\leftarrow a \\ d &\leftarrow c \end{aligned}$$

The P-stable models of $\mathcal{LP}$ are:

$$\begin{aligned} M_1 &= \{b,\ \neg a,\ \neg c,\ \neg d\}, \\ M_2 &= \{a,\ d,\ \neg b,\ \neg c\}, \\ M_3 &= \{\neg c\} \end{aligned}$$

$M_1$ and $M_2$ are also M-stable models, while $M_3$ is not (as it is a subset of the P-stable model $M_1$, as well as of $M_2$).

The M-stable models of the program $\mathcal{LP}_1$ in Example 3 are $M_1 = \{a,\ \neg b, \neg c\}$ and $M_2 = \{b,\ c,\ \neg a,\ \neg d\}$. ∎

A principal aspect of M-stable models is that they coincide with several concepts of acceptable models for a normal program.

**Proposition 3** *On a normal program $\mathcal{LP}$, Definition 6 coincides to the definitions of M-stable models of [29, 32, 31], preferred extensions of [9], the regular models of [37] and the maximal stable classes of [3].*[4]

**Proof.** Let $\mathcal{LP}$ be a normal program and $M$ be an interpretation for $\mathcal{LP}$. As a consequence of Proposition 2, $M$ is M-stable according with Definition 6 if and only if it is M-stable according with [29, 32, 31]. Therefore, the statement follows from the results in [17] and [39]. ∎

M-stable models, however, are only a first step to minimize undefinedness. The class of P-stable models which completely fulfills minimal undefinedness is that of L-stable models.

**Definition 7** *(L-stable Model)* A P-stable model $M$ is *L-stable* (*Least-undefined stable*) if the set of its undefined atoms is minimal, i.e., there exists no P-stable model $N$ of $\mathcal{LP}$ such that $\overline{N} \subset \overline{M}$. ∎

**Example 8** Consider the program $\mathcal{LP}$ of Example 7. Both M-stable models $M_1 = \{b,\ \neg a,\ \neg c,\ \neg d\}$ and $M_2 = \{a,\ d,\ \neg b,\ \neg c\}$ are also L-stable models for $\mathcal{LP}$.

For the program $\mathcal{LP}_1$ in Example 3, the M-stable model $M_2 = \{b,\ c,\ \neg a,\ \neg d\}$ is also L-stable; on the other hand, $M_1 = \{a,\ \neg b, \neg c\}$ is M-stable but it is not L-stable. ∎

[4] Note that maximal stable classes are formally defined in [38] by slightly modifying the stable classes presented in [3].

Let $\mathcal{PS}_{\mathcal{LP}}$, $\mathcal{MS}_{\mathcal{LP}}$, $LS_{\mathcal{LP}}$, and $\mathcal{TS}_{\mathcal{LP}}$ be the sets of models of a program $\mathcal{LP}$ that are P-stable, M-stable, L-stable and T-stable, respectively. We shall omit the subscript $\mathcal{LP}$ whenever $\mathcal{LP}$ is understood from the context.

**Proposition 4** *Let $\mathcal{LP}$ be a program. Then*

*(a)* $\mathcal{TS} \subseteq \mathcal{LS} \subseteq \mathcal{MS} \subseteq \mathcal{PS}$*;*

*(b)* $(\mathcal{TS} \neq \emptyset) \Rightarrow \mathcal{TS} = \mathcal{LS}$

**Proof.** *Part (a).* $\mathcal{MS} \subseteq \mathcal{PS}$ holds by Definition 6. Moreover, by Definition 7 it is clear that $\mathcal{LS} \subseteq \mathcal{MS}$. On the other hand, a T-stable model $M$ is L-stable as well, since there exists no P-stable model $N$ of $\mathcal{LP}$ such that $\overline{N} \subset \overline{M}$, as $\overline{M} = \emptyset$.
*Part (b).* Let $(\mathcal{TS} \neq \emptyset)$ and $N \in \mathcal{TS}$. From the proof of Part $(a)$ it remains to prove only that $\mathcal{TS} \supseteq \mathcal{LS}$, that is, every L-stable model is T-stable as well. Let $M \in \mathcal{LS}$. Since $N$ is also a P-stable model, from Definition 7, $\overline{N} \subset \overline{M}$ does not hold. Therefore, $\overline{M} = \emptyset$, as $\overline{N} = \emptyset$. Thus, $M$ is in $\mathcal{TS}$ as well. ∎

Thus a T-stable model is also L-stable, an L-stable model is also M-stable which is in turn P-stable, but the converse implications do not in general hold. Finally, whenever there exists a T-stable model, the definitions of T-stability and L-stability coincide.

As somehow expected, on the class of (even locally) stratified disjunctive programs, the families of P-stable models collapse into a single one.

A program $\mathcal{LP}$ is *(locally) stratified* [25] if to each ground atom $p$ of $B_{\mathcal{LP}}$ can be associated a positive integer $l(p)$ such that, for each rule $r \in \mathcal{LP}$ having $p \in H(r)$ and ground atom $q$ appearing in $r$, the following holds: (i) if $q$ appears in $H(r)$, then $l(p) = l(q)$; (ii) if $q$ appears in $B^-(r)$, then $l(p) > l(q)$; and (iii) if $q$ appears in $B^+(r)$, then $l(p) \geq l(q)$.

In particular, if all ground atoms with the same predicate have the same number associated, then $\mathcal{LP}$ is globally stratified.

The following proposition is an immediate consequence of Proposition 2 and the coincidence of 2-valued and 3-valued stable models on locally stratified programs [26].

**Proposition 5** *Let $\mathcal{LP}$ be a program. If $\mathcal{LP}$ is (locally) stratified, then* $\mathcal{PS}_{\mathcal{LP}} = \mathcal{MS}_{\mathcal{LP}} = \mathcal{LS}_{\mathcal{LP}} = \mathcal{TS}_{\mathcal{LP}}$.

**Proof.** It is known that on locally stratified programs, 2-valued and 3-valued stable models coincide [26]. Therefore, from Theorem 2, $\mathcal{TS} = \mathcal{PS}$. Thus, the statement follows from Proposition 4.(a). ∎

We close the section with a short discussion aimed at exemplifying the differences between the various stable models.

**Example 9** We have seen that the various stable model semantics differ only on unstratified programs. Unstratified negation is usually employed to specify some constraints that the stable models have to satisfy.

For instance, consider an undirected graph $G = \langle V, E\rangle$ represented by two predicates $v$ and $e$ whose instances specify the nodes and the edges of the graph, respectively, where $e$ is assumed to be symmetric. We want to individuate the partitions $\langle A, B\rangle$ of $V$, i.e. disjoint nonempty subsets $A$ and $B$ of $V$ with $A \cup B = V$, such that no edge connects any node in $A$ with any node in $B$.

The solutions to this problem can be easily found by a program using the total stable semantics. Consider the following program $\mathcal{LP}^G$.

$$
\begin{array}{ll}
r_1 : & a(X) \vee b(X) \leftarrow v(X) \\
r_2 : & abs(X) \leftarrow a(X),\ b(Y),\ e(X,Y),\ \neg abs(X) \\
r_3 : & non\_empty \leftarrow\ a(X),\ b(Y) \\
r_4 & non\_empty \leftarrow\ \neg non\_empty
\end{array}
$$

Intuitively, rule $r_1$ "guesses" a partition $\langle A, B\rangle$ of $V$ (i.e, a two sets of atoms with predicates $a$ and $b$); the (unstratified) rule $r_2$ enforces the constraint on disjointness of $\langle A, B\rangle$ [5], $r_3$ and $r_4$ enforce the constraint on nonemptiness of $\langle A, B\rangle$. It is worth noting how disjointness is enforced by $r_2$: if no node $x$ in $A$ is connected to a node $y$ in $B$ (that is, for $X = x$, the conjunction $a(X), b(Y), e(X, Y)$ is false for any instantiation of $Y$), then $A \cup B$ is a total stable model, since all instances of $r_2$ have a false body. If, on the contrary, a node $x$ in $A$ is connected to a node in $y$ in $B$, an inconsistency arises in an instance of $r_2$ (as the conjunction $a(x), b(y), e(x, y)$ is true) and $\langle A, B\rangle$ does not yield a total stable model.

Rules $r_3$ and $r_4$ enforce that the partition is not trivial: because of $r_4$, *non_empty* must be true in every stable model, and, since it must necessarily be derived from rule $r_3$, both $A$ and $B$ must contain at least one element.

Thus, every total stable model is a partition of the graph $G$ and vice versa. That is, the total stable models of $\mathcal{LP}^G$ single out all and only the solutions of the given problem.

If the problem has no solution (i.e., there exists no non-trivial partition of the nodes, which satisfies disjointness), then $\mathcal{LP}^G$ has no stable model. In this case, it can be interesting to individuate the maximal (disjoint) partial partitions of the graph, i.e., the solution of the problem where the constraint $A \cup B = V$ is relaxed to $A \cup B \subseteq V$. This is not possible with the program $\mathcal{LP}^G$ under total stable semantics. To solve this new (somewhat harder) problem, we modify rule $r_1$ of $\mathcal{LP}^G$ by adding the literal $\neg abs(X)$ in its body. The resulting program $\mathcal{LP}_1^G$ is reported below.

$$
\begin{array}{lrcl}
r_1' : & a(X) \vee b(X) & \leftarrow & v(X),\ \neg abs(X) \\
r_2' : & abs(X) & \leftarrow & a(X),\ b(Y),\ e(X,Y),\ \neg abs(X) \\
r_3' : & non\_empty & \leftarrow & a(X),\ b(Y) \\
r_4' & non\_empty & \leftarrow & \neg non\_empty.
\end{array}
$$

Under total stable semantics, nothing changes, that is, there is a one-to-one correspondence between the total stable models of $\mathcal{LP}_1^G$ and the (total) solutions of the given

[5] Note that *abs* may also be a 0-ary predicate, we have used a unary predicate to minimize the changes to the program for the next example.

problem. However, the L-stable models of $\mathcal{LP}_1^G$ allow us to identify the partial solutions of the problem. Intuitively, under partial stable semantics, if a node $x$ of $A$ is connected to some node in $B$ in the "guessed" partition, then the choice of $x$ (for $A$) is "annulled" by the presence of $\neg abs(x)$ in $r_1'$ which is necessarily undefined (because of rule $r_2'$). Thus, every P-stable model $M$ represents a (possibly empty) disjoint partition of the graph, given by the atoms on $a$ and $b$ that are true in $M$. Let us see this in more detail. Let $M$ be a P-stable model. To show that $M$ is a disjoint partition, we have to demonstrate that if $a(x)$ is in $M$, then $e(x,y)$ does not hold, for any $y$ such that $b(y) \in M$ (that is, $x \in A$ is not connected to any $y \in B$). Now, if $a(x) \in M$, then $abs(x)$ must necessarily be false, and, as a consequence, no $y$ is such that both $b(y)$ and $e(x,y)$ are true, since $abs(x)$ could not be false in the latter case. Hence, for each $x \in A$ (i.e., s.t. $a(x) \in M$), there does not exist $y \in B$ (i.e., s.t. $b(x) \in M$) such that there is an edge from $a$ to $b$. Therefore, $M$ represents a disjoint partition of the graph (note that, since the $e$ predicate is symmetric, we do not need to prove that there is no edge from $B$ to $A$). Therefore, every P-stable model $M$ represents a disjoint partition of the graph and, in particular, every L-stable model $M$ represents a *maximal* disjoint partition of the graph.

On the other hand, it is soon recognized that if $\langle A, B\rangle$ is a maximal disjoint partition, then the interpretation

$$\begin{aligned} M = \ & \{a(x) \mid x \in A\} \cup \{b(x) \mid x \in B\} \cup \{\neg b(x) \mid x \notin A\} \cup \{\neg a(x) \mid x \in B\} \cup \\ & \{\neg abs(x) \mid x \in A \cup B\} \cup N, \end{aligned}$$

is an L-stable model of the program, where $N$ is either $\emptyset$ or $\{non\_empty\}$ according whether $\langle A, B\rangle$ is trivial or not.

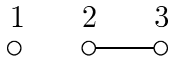
Since, by virtue of Proposition 4.(b), L-stable semantics differs from total stable semantics *only if* there is no total stable models, we have that L-stable semantics shows the following behavior:

- If the problem admits some "exact" solution (i.e., graph $G$ admits some total partition), then there is a one-to-one correspondence between the L-stable models of $\mathcal{LP}_1^G$ and the total partitions of the graph.
- If the problem admits no "exact" solution (i.e., graph $G$ admits no total partition), then, there is a one-to-one correspondence between the L-stable models of $\mathcal{LP}_1^G$ and the maximal partial partitions of the graph. (Note that in case the problem admits only trivial partitions, $\mathcal{LP}_1^G$ has only some L-stable models representing them, where *non_empty* is not true).

Thus, L-stable semantics gives exactly what one is looking for, since we are interested in partial partitions (approximate solutions) *only if* there is no total partition (as in presence of exact total solutions of the problem, we would like that the program singles out only them).

In other words, L-stable semantics allows us to find "approximate" (i.e., maximal partial) solutions of the problem in case where the problem admits no exact solution, giving the exact solutions if they exist.

It is worth observing the behavior of M-stable semantics in this case. Every M-stable model of $\mathcal{LP}_1^G$ is a partition of the graph but, differently from L-stable models, it is not necessarily a *maximal* partition. The problem is that it may happen that a P-stable model describes a partition which can not be extended to a maximal partition. For instance, consider the following graph:



The graph has three vertices 1–3, and only one edge, which connects 2 and 3. Then, the following interpretation is a P-stable model of the program:

$$M = \{a(1), \neg b(1), \neg abs(1), \neg a(2), b(2), \neg abs(2), \neg b(3), non\_empty\}$$

This corresponds to the partition $A = \{1\}$ and $B = \{2\}$. Now, it is not possible to extend this model M by adding atom $b(3)$, to arrive at a model corresponding to the partition $A = \{1\}, B = \{2, 3\}$ which is maximal (note that this is possible with L-stable semantics).

The problem is that the 'b-atoms' which are not in the current set $B$, must be false and hence can not be included in a larger P-stable model. Thus, in order to obtain the maximal partitions under M-stable semantics we have to prevent that 'b-atoms' which are not in the current set $B$ are made false even if they are not in the set $A$. To this end, consider the program $\mathcal{LP}_2^G = \mathcal{LP}_1^G \cup \{b(X) \leftarrow abs(X)\}$. Due to the inclusion of the latter rule, if a node $x$ is neither in $A$ nor in $B$, then both $a(x)$ and $b(x)$ are undefined in every P-stable model of $\mathcal{LP}_2^G$ (note that $b(x)$ is false in the P-stable models of $\mathcal{LP}_1^G$). In this way, if a P-stable model $M$ of $\mathcal{LP}_2^G$ describes a partition which is not maximal, then $M$ is not M-stable, as it can be extended to a larger stable model which represents the maximal partition.

Hence, the maximal partitions of the graph can be singled out also by M-stable semantics, but the appropriate program is less intuitive. Moreover, in case that the problem admits total partitions (i.e., $\mathcal{LP}_2^G$ admits total stable models), differently from the L-stable semantics, M-stable semantics gives also the partial maximal partitions of the graph (which are unwanted in this case). ∎

Example 9 confirms our claim that L-stable model is the natural relaxation of the notion of total stable model and that L-stable semantics is the best way to "approximate" T-stable semantics. One may now ask why we presented M-stable models also. The reason for that is mainly twofold:

- First, on normal programs the concept of M-stable model turned out to be important, since it coincides with several proposals for the intended models of a program, which have been for formulated in quite different settings [3, 9, 29, 32, 31, 39]. Thus, the generalization of this concept to the class of disjunctive programs is natural, similar as the generalization of stable models or regular models for normal programs to disjunctive programs [26, 38].

- Second, as shown in Section 6.1, M-stable models present some desirable modularity properties which make them amenable from the methodological and computational point of view, especially in the context of deductive databases.

# 5 Comparison with Regular Models

The notion of regular model [37, 38] is similar in spirit to M-stable model, but is based on a weaker concept of model than P-stable model.

**Definition 8** *(Regular Model)[38]* Let $\mathcal{LP}$ be a program and $M$ an interpretation of it. $M$ is a *regular model* of $\mathcal{LP}$ if the following conditions are satisfied.

$(a)$ $M$ is a 3-valued model of $\mathcal{LP}$,

$(b)$ $M$ is founded, and

$(c)$ no founded 3-valued model $N$ of $\mathcal{LP}$ such that $N \supset M$. ∎

(Observe that Condition $(b)$ is called *justifiability* in [37, 38].) Thus, regular models are very close to M-stable models. At a first glance, one can even think that the two definitions coincide: both regular and M-stable are founded 3-valued models, and maximal under set inclusion. This is in fact the case for normal programs (cf. Proposition 3). However, a closer look to the definitions reveals that Definition 8 is different from Definition 6. Let $\mathcal{RS}_{\mathcal{LP}}$ and $\mathcal{FS}_{\mathcal{LP}}$ denote the family of regular and founded 3-valued models of $\mathcal{LP}$, respectively. In general, we have the following relation between $\mathcal{RS}_{\mathcal{LP}}$ and $\mathcal{MS}_{\mathcal{LP}}$.

**Proposition 6** *Let $\mathcal{LP}$ be a program. Then, for every M-stable model $M \in \mathcal{MS}_{\mathcal{LP}}$ there exists a regular model $N \in \mathcal{RS}_{\mathcal{LP}}$ such that $M \subseteq N$.*

**Proof.** By Definition 6, every M-stable is a maximal element in $\mathcal{PS}$. On the other hand, by Definition 3, every P-stable is founded and, by virtue of Theorem 1, is also a 3-valued model. Therefore, $\mathcal{PS} \subseteq \mathcal{FS}$. Since M-stable are maximal elements of $\mathcal{PS}$ and regular models are maximal elements of $\mathcal{FS}$, the statement follows. ∎

Proposition 6 does not exclude the possibility that the $\mathcal{RS} = \mathcal{MS}$ (actually, as we will see later, the equality holds for a meaningful class of programs). However, the difference between the two families is soon recognized by the observation that every program admits a regular model [38], while there are programs that lack M-stable models (this is an immediate consequence of the fact that some programs miss 3-valued stable models [26]). The following example from [26] makes this point clear.

**Example 10** Let $\mathcal{LP}$ be the program below.

$$\begin{array}{l} work \vee sleep \vee tired \leftarrow \\ work \leftarrow \neg tired \\ sleep \leftarrow \neg work \\ tired \leftarrow \neg sleep \end{array}$$

It is easy to see that $\mathcal{LP}$ has no P-stable model (cf. [26]), and hence no M-stable and L-stable model. On the other hand, $\{work\}$, $\{sleep\}$, and $\{tired\}$ are (all the) regular models of $\mathcal{LP}$. ∎

At a first glance, the behavior of regular semantics can appear even more intuitive than that of M-stable semantics. However, as announced in the introduction, this behavior is somehow the proof that regular models do not respect the closed world assumption. The *closed world assumption* (*CWA*), introduced by Reiter in [27], informally states that an atom must be assumed false if it is not entailed from the theory. In logic programming the first explicit CWA rule, called *Generalized CWA* (*GCWA*), is due to Minker [23]. According with GCWA, if an atom $a$ is false in every (total) minimal model of the program, then $\neg a$ is assumed true. The closed world assumption has been the inspiring principle of the semantics of negation in logic programming and led to minimize the set of positive conclusions maximizing the negative ones. Since the semantics of a program $\mathcal{LP}$ is usually identified by a set of "preferred" models for $\mathcal{LP}$ (e.g., the set of stable models or the well-founded model), which can be seen as alternative meanings of $\mathcal{LP}$, in the framework of normal programs, CWA can be restated as the following property that each preferred model $M$ of a given semantics should satisfy:

> "If every rule with $A$ in the head has a false body w.r.t. $M$, then $A$ must be false in $M$."

It is easy to see that (at least in this weak form) the CWA is satisfied by many well-known semantics for datalog. For instance, stratified models [2], Fitting's models [15], well-founded models [36], both total and partial stable models [16, 26], and regular models [37] for normal programs satisfy the principle of CWA. Since it is widely acknowledged that inferences from disjunctive rules must follow a minimality criterion, the CWA can be suitably extended to disjunctive programs as follows:

> "If every rule with $A$ in the head has a false body or contains in the head an atom different from $A$ that is true w.r.t. $M$, then $A$ must be false in $M$."

Minker's minimal models [23] and both total and partial stable models [26] for disjunctive programs satisfy the CWA. As a consequence, both L-stable and M-stable models satisfy the CWA as well. On the contrary, regular models for disjunctive programs do not satisfy the CWA. To see the latter point, observe the above example and consider its regular model $M = \{sleep\}$: There are two rules with *tired* in the head, the former ($work \vee sleep \vee tired \leftarrow$) contains the true atom *sleep* in the head, the other rule ($tired \leftarrow \neg sleep$) has a false body. Thus, according with the CWA rule, *tired* should be false in $M$; nevertheless, this atom is undefined. Hence, the existence of regular models for disjunctive programs which do not have M-stable models may be explained by the violation of the above closed world assumption by such regular models. Moreover, it is important to point out that regular models may not agree with M-stable models also for programs where M-stable models do exist.

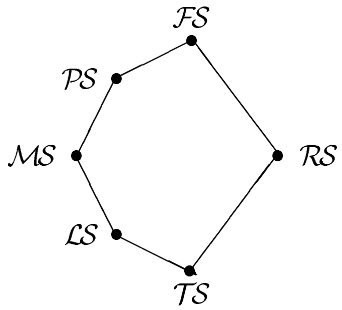



Figure 1: Relation between partial model semantics

**Example 11** Consider the following extension $\mathcal{LP}'$ of the program shown in Example 10.

$$
\begin{array}{l}
adult \vee child \leftarrow \\
work \vee sleep \vee tired \leftarrow adult \\
work \leftarrow \neg tired,\ adult \\
sleep \leftarrow \neg work,\ adult \\
tired \leftarrow \neg sleep
\end{array}
$$

$\mathcal{LP}'$ has a unique total stable model $M = \{child,\ \neg adult,\ \neg sleep,\ tired,\ \neg work\}$, which is also the unique P-stable, M-stable and L-stable model. Nevertheless, $\mathcal{LP}'$ admits besides $M$ three other regular models, namely $\{adult,\ \neg child,\ sleep\}$, $\{adult,\ \neg child,\ work\}$, and $\{adult,\ \neg child,\ tired\}$. ∎

From Example 11, we derive that, differently from L-stable semantics, even if the program has some T-stable model, regular semantics does not necessarily agree with T-stable semantics.

**Proposition 7** *Let $\mathcal{LP}$ be a program. Then, $\mathcal{TS} \neq \emptyset \not\Rightarrow \mathcal{TS} = \mathcal{RS}$.*

In Example 11, where a unique total stable model exists, regular semantics does not coincide to total stable semantics, while P-stable and M-stable and L-stable do coincide. Unfortunately, for P-stable and M-stable models this desirable behavior cannot be generalized to all cases of existence of total stable models. On the contrary, as shown in Proposition 4.(b), L-stable models do always coincide to total stable models if some total stable exists. This property is very important, as it is generally acknowledged that if total stable models exist, they are the natural meaning of the program, and candidates L-stable models as the best "approximation" of total stable models.

The Hasse diagram of the inclusion order between different sets of partial models is displayed in Figure 1.

Thus, in general, the concepts of models are different. The two following propositions identify simple classes of programs where the models coincide. An immediate result is that all families of models coincide on the class of stratified programs.

**Proposition 8** *Let $\mathcal{LP}$ be a stratified program. Then $\mathcal{PS}_{\mathcal{LP}} = \mathcal{MS}_{\mathcal{LP}} = \mathcal{LS}_{\mathcal{LP}} = \mathcal{TS}_{\mathcal{LP}} = \mathcal{RS}_{\mathcal{LP}}$.*

**Proof.** From [38], on stratified programs we have that $\mathcal{TS}_{\mathcal{LP}} = \mathcal{RS}_{\mathcal{LP}}$. Thus, the statement follows immediately from Proposition 5. ∎.

Another known equivalence between M-stable and regular models holds for normal programs as already mentioned in Proposition 3.

**Proposition 9** *[39] Let $\mathcal{LP}$ be a normal program. Then $\mathcal{MS}_{\mathcal{LP}} = \mathcal{RS}_{\mathcal{LP}}$.*

From the results of Propositions 8 and 9, a natural question arises: is it possible to enlarge the class of normal programs on the one hand and stratified disjunctive programs on the other, by possibly combining such programs, ensuring that $\mathcal{MS} = \mathcal{RS}$ still holds? An answer to this issues is provided by corollaries to the results in the next section. Informally, if we start from a normal or disjunctive program, and impose on top a (another) normal program that uses the relations computed by the "lower" program like input relations, then M-stable and regular models coincide. Notice that this class greatly extends stratified disjunctive programs to programs that allow for unstratified rules.

# 6 Modularity Property of Partial Semantics

In this section we first derive some modularity properties for partial semantics. Then, we apply the results to enlarge the class of programs on which M-stable and regular models are known to coincide. Moreover, we also derive useful computational properties for for both M-stable and regular models. Similar modularity properties for normal programs have been considered in [34, 33, 28] and, for disjunctive programs, in [7] and [11].

Let $\mathcal{LP}_1$ and $\mathcal{LP}_2$ be programs. We write $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$ iff no predicate occurring in $\mathcal{LP}_1$ occurs in the head of any rule of $\mathcal{LP}_2$ [11]. Moreover, given an interpretation $M$ for $\mathcal{LP}_1$, we denote by $\mathcal{LP}_2^M$ the program

$$\mathcal{LP}_2 \cup \{a \leftarrow \ \mid a \in M^+\} \cup \{a \leftarrow \neg a \mid a \in \overline{M}\} \cup \{a \leftarrow a \mid \neg a \in M^-\}$$

(note that both $M^+$ and $\overline{M}$ are subsets of $B_{\mathcal{LP}_1}$). Intuitively, if $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$, then every atom occurring in the head of a rule in $\mathcal{LP}_2$ has no influence on any atom from $\mathcal{LP}_1$, and cannot be used for any derivation in $\mathcal{LP}_1$. The rules added to $\mathcal{LP}_2$ in order to construct $\mathcal{LP}_2^M$ provide the interpretation $M$ as input to $\mathcal{LP}_2$, where P-stable (and regular) semantics assures that the value of the atoms from $\mathcal{B}_{\mathcal{LP}_1}$ in a P-stable (resp. regular) model of $\mathcal{LP}_2^M$ coincides with its value in $M$. (In particular, the rules $a \leftarrow a$) serve to guarantee that atom $a$ belongs to $\mathcal{B}_{\mathcal{LP}_2^M}$, and is assigned false as in $M$.)

**Example 12** Consider $\mathcal{LP}_1$ and $\mathcal{LP}_2$:

$$\mathcal{LP}_1: \quad \begin{array}{r} d \vee b \leftarrow a \\ a \leftarrow \neg b \end{array} \qquad\qquad \mathcal{LP}_2: \quad \begin{array}{l} e \leftarrow d, \neg e \\ f \leftarrow d \end{array}$$

Here, $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$. For $M = \{a, \neg b\}$, we have $\mathcal{LP}_2^M = \mathcal{LP}_2 \cup \{\, a \leftarrow,\; b \leftarrow b,\; d \leftarrow \neg d \,\}$. ∎

The next proposition tells us that the regular models of a program $\mathcal{LP}$, which consists of a normal program $\mathcal{LP}_2$ on top of an arbitrary disjunctive program $\mathcal{LP}_1$ can be built bottom up, i.e., by first building the regular models of $\mathcal{LP}_1$ and then by extending each such model $M$, considered like input data, to regular models of $\mathcal{LP}$ by computing the regular models of $\mathcal{LP}_2^M$. This yields all and only the regular models for $\mathcal{LP}$. A similar result has been reported in [11] for T-stable models.

**Proposition 10** *Let $\mathcal{LP}_1$ and $\mathcal{LP}_2$ be programs such that $\mathcal{LP}_2$ is a normal program and $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$. Let $\mathcal{LP} = \mathcal{LP}_1 \cup \mathcal{LP}_2$. Then,*

1. *for each $M \in \mathcal{RS}_{\mathcal{LP}}$, $M \cap Lit_{\mathcal{LP}_1} \in \mathcal{RS}_{\mathcal{LP}_1}$.*
2. *$\mathcal{RS}_{\mathcal{LP}} = \cup_{M_1 \in \mathcal{RS}_{\mathcal{LP}_1}} \mathcal{RS}_{\mathcal{LP}_2^{M_1}}$.*

This property is somewhat surprising. It is intuitive that the regular models of $\mathcal{LP} = \mathcal{LP}_1 \cup \mathcal{LP}_2$ must be obtained by extending founded models $M_1$ of $\mathcal{LP}_1$ to regular models of $\mathcal{LP}_2$; however, due the principle of minimal undefinedness, which endows in maximization of founded models, it seems that one loses regular models by extending only regular, i.e. maximal founded models $M_1$ of $\mathcal{LP}_1$ to regular models of $\mathcal{LP}_2$.

Before providing the formal proof of Proposition 10, we demonstrate the proposition on the previous example.

**Example 13** Let $\mathcal{LP} = \mathcal{LP}_1 \cup \mathcal{LP}_2$, where $\mathcal{LP}_1$ and $\mathcal{LP}_2$ are the programs reported in Example 12. The regular models of $\mathcal{LP}$ are: $M_1 = \{\neg b, a, d, f\}$ and $M_2 = \{\neg d, \neg f, \neg e\}$. The hypotheses of Proposition 10 are verified, as $\mathcal{LP}_2$ is normal and $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$. Indeed, according with Proposition 10.1, both $M_1' = M_1 \cap Lit_{\mathcal{LP}_1} = \{\neg b, a, d\}$ and $M_2' = M_2 \cap Lit_{\mathcal{LP}_1} = \{\neg d\}$ are regular models for $\mathcal{LP}_1$. Moreover, $M_1'$ and $M_2'$ are the only regular models for $\mathcal{LP}_1$, and, according with Proposition 10.2, we have that $\mathcal{RS}_{\mathcal{LP}} = \mathcal{RS}_{\mathcal{LP}_2^{M_1'}} \cup \mathcal{RS}_{\mathcal{LP}_2^{M_2'}}$. Indeed, $M_1$ is the only regular model of $\mathcal{LP}_2^{M_1'} = \mathcal{LP}_2 \cup \{a \leftarrow\ ,\; d \leftarrow,\; b \leftarrow b\}$, and $M_2$ is the only regular model of $\mathcal{LP}_2^{M_2'} = \mathcal{LP}_2 \cup \{a \leftarrow \neg a,\; b \leftarrow \neg b,\; d \leftarrow d\}$. ∎

We next provide the formal proof of the statement.

**Proof.** (of Proposition 10)
*part 1.* Let $M_1 = M \cap Lit_{\mathcal{LP}_1}$. We have to show that $M_1$ is a regular model of $\mathcal{LP}_1$, i.e.,

(i) $M_1$ is a founded model of $\mathcal{LP}_1$, and (ii) no founded model $N_1$ of $\mathcal{LP}_1$ exists such that $N_1 \supset M_1$.

(i) $M_1$ *is a founded model.* $M_1$ is a 3-valued model for $\mathcal{LP}_1$, as $ground(\mathcal{LP}_1) \subseteq ground(\mathcal{LP})$ and a rule in $ground(\mathcal{LP}_1)$ is satisfied w.r.t. $M_1$ iff it is satisfied w.r.t. $M$ (since no literal appearing in $ground(\mathcal{LP}_1)$ occurs in $M - M_1$). By deriving a contradiction, we show that $M_1$ is founded. Assume that $M_1$ is not founded for $\mathcal{LP}_1$. Hence, there exists a total model $M_2$ of $\mathcal{LP}_1(M_1)$ such $M_2^+ \subset M_1^+$. Since no literal appearing in $ground(\mathcal{LP}_1)$ occurs in $M - M_1$, $\mathcal{LP}(M) = \mathcal{LP}_1(M_1) \dot{\cup} \mathcal{LP}_2(M)$, where $\dot{\cup}$ denotes disjoint union. Therefore, the total interpretation $N$ of $\mathcal{LP}(M)$ such that $N^+ = M_2^+ \cup (M^+ - M_1^+)$ is a model of $\mathcal{LP}(M)$ (notice that this program is positive). However, $N^+ \subset M^+$, and hence no $M' \in \mathrm{MM}(\mathcal{LP}(M))$ exists such that $M'^+ = M^+$; this implies that $M$ is not a founded model of $\mathcal{LP}$, and we reached a contradiction. Consequently, $M_1$ is a founded model of $\mathcal{LP}_1$.

(ii) $\not\exists N_1 \supset M_1$ *s.t.* $N_1$ *is a founded model for* $\mathcal{LP}_1$. We proceed by contradiction. Let $N_1$ be a founded model for $\mathcal{LP}_1$ such that $N_1 \supset M_1$. We demonstrate that this implies the existence of a founded model $N$ of $\mathcal{LP}$ such that $N \supset M$, contradicting the fact that $M$ is a regular model of $\mathcal{LP}$.

Let $M_2$ be the (unique) minimal (total) model of the program $\mathcal{LP}_0 = \mathcal{LP}_2^{N_1}(M \cup N_1)$, and choose $N = M \cup N_1 \cup M_2^+$. Observe that $\mathcal{LP}_0$ is a positive normal program, which coincides with $\mathcal{LP}_2^{N_1}(N)$.

Clearly, $N \supset M$ (notice that $N_1 - M \neq \emptyset$). Moreover, since $N_1$ is a founded model of $\mathcal{LP}_1$ and $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$, in order to show that $N$ is a founded model of $\mathcal{LP}$ it suffices to show that $N$ is a founded interpretation of $\mathcal{LP}_0$ and that $N$ is a model of $\mathcal{LP}$.

First, we show that $N$ is an interpretation of $\mathcal{LP}$, i.e., a consistent set of ground literals from $\mathcal{B}_{\mathcal{LP}}$. It is easy to see that $M \cup N_1$ is consistent, since $M \cap Lit_{\mathcal{LP}_1} = M_1 \subset N_1$. Thus, to prove the consistency of $N$, it suffices to show the following:

*Claim*: No atom $a$ from $M_2^+$ is false in $M \cup N_1$.

The claim is proven by induction on the derivation of $a$ from $\mathcal{LP}_0$ using the immediate consequence operator $T_{\mathcal{LP}_0}(\cdot)$. (Recall that $M_2^+ = T^{\infty}_{\mathcal{LP}_0}$.) Assume that $a$ is obtained in the $i$-th step.

(*Base*) $i = 1$: Hence, the rule $a \leftarrow$ is in $ground(\mathcal{LP}_0)$. If $a \in B_{\mathcal{LP}_1}$, then $a \in N_1$ and hence $a$ is not false in $M \cup N_1$. Thus, assume that $a \notin B_{\mathcal{LP}_1}$. This implies that there exists some rule $r$ in $ground(\mathcal{LP}_2)$ where $H(r) = a$, $B^+(r) = \emptyset$, and $B^-(r)$ is true w.r.t $M \cup N_1$. Consequently, each literal $\neg b \in B^-(r)$ has in $M$ either value true or undefined. Since $M$ is a model of $\mathcal{LP}_2$, it follows that $a$ is not false in $M$. Thus, $a$ is not false in $M \cup N_1$.

(*Induction*) $i > 1$: $a$ is derived from a rule $H(r) \leftarrow B^+(r)$ in $ground(\mathcal{LP}_0)$, for some rule $r$ from $ground(\mathcal{LP}_2)$ where $H(r) = a$, $B^+(r) \neq \emptyset$ and $B^-(r)$ is true in $M \cup N_1$. From the inductive hypothesis, we have that no atom in $B^+(r)$ is false in $M \cup N_1$. Thus, it follows that $B(r)$ is not false in $M$. Since $M$ is a model of $\mathcal{LP}_2$, it follows that $a$ is not false in $M$. This proves the claim.

Thus, we have shown that $N$ is an interpretation of $\mathcal{LP}$. We next check that $N$ is founded w.r.t. $\mathcal{LP}_0$, i.e., $N^+ = A^+$ for some $A \in \mathrm{MM}(\mathcal{LP}_0(N))$.

We have that

$$\mathcal{LP}(N) = \mathcal{LP}_1(N)\dot{\cup}\mathcal{LP}_2(N) = \mathcal{LP}_1(N_1)\dot{\cup}\mathcal{LP}_2(N).$$

Since $N_1$ is a founded model of $\mathcal{LP}_1$, $N_1^+ = L_1^+$ for some $L_1 \in \mathrm{MM}(\mathcal{LP}_1(N_1))$. From the definition of $\mathcal{LP}_0$, it follows that $M_2 \cap B_{\mathcal{LP}_1} = N_1^+$, and since $M_2$ is the (unique) minimal model of $\mathcal{LP}_0$, it follows that $M_2^+ = L_2^+$ for some $L \in \mathrm{MM}(\mathcal{LP}(N)) = \mathrm{MM}(\mathcal{LP}_1(N_1)$ $\dot{\cup}\mathcal{LP}_2(N))$. It holds that $M_2^+ \supseteq M^+ \cup N_1^+$: Clearly, from the construction of $\mathcal{LP}_0$ and $\mathcal{LP}_0 = \mathcal{LP}_2^{N_1}(N)$, we have $N_1^+ \subseteq M_2^+$. On the other hand, since $M$ is a founded model of $\mathcal{LP}$ and $\mathcal{LP}_2 \rhd \mathcal{LP}_1$, the normal positive program $\mathcal{LP}_3 = \mathcal{LP}_2(M) \cup \{a \leftarrow \mid a \in M_1^+\}$ has a unique total minimal model $M'$, which satisfies $M'^+ = M^+$. Since $\mathcal{LP}_3 \subseteq \mathcal{LP}_0$ and $\mathcal{LP}_0$ is normal (and positive), it follows $M^+ = M'^+ \subseteq M_2^+$. Thus, we have $M_2^+ \supseteq M^+ \cup N_1^+$. This means $N^+ = M^+ \cup N_1^+ \cup M_2^+ = M_2^+$. This proves that $N$ is founded w.r.t. $\mathcal{LP}_0$.

It thus remains to show that $N$ is a 3-valued model of $\mathcal{LP}$. Clearly, $N$ satisfies each rule in $ground(\mathcal{LP}_1)$. Consider now any rule $r$ from $ground(\mathcal{LP}_2)$. We show that $r$ is satisfied by $N$, by checking that $r$ cannot be violated. For that, we need to check the cases where (1) $H(r)$ is false in $N$, or (2) $H(r)$ is undefined in $N$.

(1) If $H(r)$ is false in $N$, then $H(r)$ is false also in $M$. Since $M$ satisfies $r$, $B(r)$ is false in $M$. Thus, it follows that $B(r)$ is also false in $N$ as $N \supseteq M$. (2) If $H(r)$ is undefined in $N$, then $B(r)$ is not true in $N$. To show this, by contradiction suppose that $B(r)$ is true in $N$. Then, $B^-(r)$ is true in $M \cup N_1$, and hence $H(r) \leftarrow B^+(r)$ belongs to $ground(\mathcal{LP}_0)$. Since $B^+(r)$ is true in $N$ and, as shown already above, $N^+ = M_2^+$, it follows that $B^+(r)$ is true in $M_2$, and thus $H(r)$ is true in $M_2$. By definition of $N$, it thus follows that $H(r)$ is true in $N$. Thus we arrived at a contradiction.

We thus have that $N$ is a model of $\mathcal{LP}$. Now the founded model $N$ of $\mathcal{LP}$ raises a contradiction to the fact that $M$ is a regular model of $\mathcal{LP}$. This concludes the proof of part 1.

*part 2.* ($\subseteq$) Let $M \in \mathcal{RS}_{\mathcal{LP}}$. Then, by part 1, $M_1 = M \cap Lit_{\mathcal{LP}_1}$ is a regular model of $\mathcal{LP}_1$. Suppose $M$ is not a regular model of $\mathcal{LP}_2^{M_1}$. Clearly, $M$ is a founded model of $\mathcal{LP}_2^{M_1}$. Hence, there exists a founded model $N$ of $\mathcal{LP}_2^{M_1}$ such that $N \supset M$. From the definition of $\mathcal{LP}_2^{M_1}$ and the regularity of $M_1$, it follows that $N$ is a founded model of $\mathcal{LP}$. Thus, $M$ is not a regular model of $\mathcal{LP}$; we reached a contradiction. This proves the "$\subseteq$" inclusion. ($\supseteq$) Let $N$ be a regular model of $\mathcal{LP}_2^{M_1}$ for some regular model $M_1$ of $\mathcal{LP}_1$. Suppose that $N$ is not a regular model of $\mathcal{LP}$. Then, it follows that there exists a founded model $N'$ of $\mathcal{LP}$ such that $N' \supset N$. In particular, $N' \cap Lit_{\mathcal{LP}_1} = M_1$, and as easy to check, $N'$ is a founded model of $\mathcal{LP}_2^{M_1}$. This means that $N$ is not a regular model of $\mathcal{LP}_2^{M}$, however, and we thus reached a contradiction. This proves the "$\supseteq$" inclusion and completes the proof of the proposition. ∎

The restriction of $\mathcal{LP}_2$ in Proposition 10 to normal programs is essential. If we allow disjunctive rules in $\mathcal{LP}_2$, then the proposition fails. This is shown by the following example.

$$\mathcal{LP}_1: \quad \begin{array}{l} b \leftarrow \neg a \\ a \leftarrow \neg b \end{array} \qquad \mathcal{LP}_2: \quad \begin{array}{r} d \vee e \leftarrow c \\ c \leftarrow b \\ c \leftarrow a \end{array}$$

Clearly $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$. The interpretation $M = \{\neg d\}$ is a regular model of $\mathcal{LP} = \mathcal{LP}_1 \cup \mathcal{LP}_2$. However, $M \cap Lit_{\mathcal{LP}_1} = \emptyset$ is not a regular model of $\mathcal{LP}_1$. In fact, the regular models of $\mathcal{LP}_1$ are $M_1 = \{a, \neg b\}$ and $M_2 = \{\neg a, b\}$.

For arbitrary $\mathcal{LP}_2$, the following weaker form of Proposition 10 holds; we leave the proof, which can be extracted from the proof of Proposition 10, to the reader. (Recall that $\mathcal{FS}_{\mathcal{LP}}$ denotes the founded models for $\mathcal{LP}$.)

**Proposition 11** *Let $\mathcal{LP}_1$ and $\mathcal{LP}_2$ be programs such that $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$. Let $\mathcal{LP} = \mathcal{LP}_1 \cup \mathcal{LP}_2$. Then,*

1. *for each $M \in \mathcal{RS}_{\mathcal{LP}}$, $M \cap Lit_{\mathcal{LP}_1} \in \mathcal{FS}_{\mathcal{LP}_1}$.*
2. *$\mathcal{RS}_{\mathcal{LP}} = \max_{\subseteq}\left(\cup_{M_1 \in \mathcal{FS}_{\mathcal{LP}_1}} \mathcal{RS}_{\mathcal{LP}_2^{M_1}}\right)$.*

Here $\max_{\subseteq}$ denotes the maximal sets w.r.t. inclusion.

With a similar proof, we can derive that the modularity property for regular models holds for M-stable models as well.

**Proposition 12** *Let $\mathcal{LP}_1$ and $\mathcal{LP}_2$ be programs such that $\mathcal{LP}_2$ is a normal program and $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$. Let $\mathcal{LP} = \mathcal{LP}_1 \cup \mathcal{LP}_2$. Then,*

1. *for each $M \in \mathcal{MS}_{\mathcal{LP}}$, $M \cap Lit_{\mathcal{LP}_1} \in \mathcal{MS}_{\mathcal{LP}_1}$.*
2. *$\mathcal{MS}_{\mathcal{LP}} = \cup_{M_1 \in \mathcal{MS}_{\mathcal{LP}_1}} \mathcal{MS}_{\mathcal{LP}_2^{M_1}}$.*

**Proof.** (Outline) The proof is similar to the proof of Proposition 10. For part 1, one shows that $M_1 = M \cap Lit_{\mathcal{LP}_1}$ is a P-stable model of $\mathcal{LP}_1$, and then shows by deriving a contradiction that no larger P-stable model $N_1 \supset M_1$ exists. Along the lines of the proof of Proposition 10, one constructs the founded model $N \supset M$ of $\mathcal{LP}$. Clearly, $N$ is a founded model of $\mathcal{LP}_2^{N_1}$. Thus, a regular model $N' \supseteq N$ of $\mathcal{LP}_2^{N_1}$ exists; notice that $N' \cap Lit_{\mathcal{LP}_1} = N_1$. Since $\mathcal{LP}_2^{N_1}$ is a normal program, $N'$ is an M-stable model of $\mathcal{LP}_2^{N_1}$ by Proposition 9. Thus, as easily checked, $N'$ is a P-stable model of $\mathcal{LP}$. $N$ contradicts the fact that $M$ is an M-stable model of $\mathcal{LP}$; this proves part 1. Part 2 is shown analogous as in Proposition 10. ∎

Analogously, the weak form of modularity in Proposition 11, with $\mathcal{RS}$ and $\mathcal{FS}$ replaced by $\mathcal{MS}$ and $\mathcal{PS}$, respectively, holds for M-stable models.

We remark that for P-stable models, we obtain a stronger result on modularity than the one for M-stable models. In fact, the P-stable models of $\mathcal{LP} = \mathcal{LP}_2 \cup \mathcal{LP}_1$ with $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$ are obtained by extending all P-stable models of $\mathcal{LP}_1$ to P-stable models of $\mathcal{LP}_2$.

**Proposition 13** *Let $\mathcal{LP}_1$ and $\mathcal{LP}_2$ be programs such that $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$. Let $\mathcal{LP} = \mathcal{LP}_1 \cup \mathcal{LP}_2$. Then,*

1. *for each $M \in \mathcal{PS}_{\mathcal{LP}}$, $M \cap Lit_{\mathcal{LP}_1} \in \mathcal{PS}_{\mathcal{LP}_1}$.*
2. *$\mathcal{PS}_{\mathcal{LP}} = \cup_{M_1 \in \mathcal{PS}_{\mathcal{LP}_1}} \mathcal{PS}_{\mathcal{LP}_2^{M_1}}$.*

**Proof.** Omitted. ∎

As an immediate corollary of Proposition 13, we obtain that the same holds for T-stable models; the latter was reported in [11].

It is worth noting that Proposition 10 fails for L-stable models in place of regular models. This is not surprising, since a P-stable model might not be extendible to an L-stable model by adding literals. Consider the following example.

$$
\begin{array}{llcll}
\mathcal{LP}_1 : & b \leftarrow \neg a & \qquad & \mathcal{LP}_2 : & d \leftarrow a \\
 & a \leftarrow \neg b & & & d \leftarrow \neg a, \neg d \\
 & c \leftarrow a, \neg c & & &
\end{array}
$$

Here, the P-stable models of $\mathcal{LP}_1$ are $\emptyset$, $\{a, \neg b\}$, and $\{\neg a, b, \neg c\}$; the last one of them is L-stable. The P-stable models of $\mathcal{LP} = \mathcal{LP}_1 \cup \mathcal{LP}_2$ are $M_1 = \emptyset$, $M_2 = \{a, \neg b, d\}$, and $M_3 = \{\neg a, b, \neg c\}$, from which the latter two are L-stable. Now $\{a, \neg b, d\} \cap Lit_{\mathcal{LP}_1} = \{a, \neg b\}$ is not an L-stable model of $\mathcal{LP}_1$. Thus, the condition 1.) fails. Moreover, $M_2$ is an L-stable model of $\mathcal{LP}$, but $M_2$ is not an L-stable model of $\mathcal{LP}_2^{\{\neg a, b, \neg c\}}$; this shows that also condition 2.) fails.

However, for L-stable models we have weaker properties, similar to the the properties of M-stable models for general $\mathcal{LP}_1$ and $\mathcal{LP}_2$. The L-stable models of $\mathcal{LP} = \mathcal{LP}_1 \cup \mathcal{LP}_2$ such that $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$ are obtained by the least undefined models in the collection of all P-stable models of $\mathcal{LP}_2^{M_1}$, where $M_1$ is a P-stable of $\mathcal{LP}_1$. Notice that if $\mathcal{LP}_1$ is stratified, then each L-stable model of $\mathcal{LP}$ is total on $\mathcal{LP}_1$.

### 6.1 Application of Modularity

The modularity property of M-stable and regular models has interesting applications. First, it allows to identify an extended class of programs on which regular and M-stable semantics coincide. Second, an interesting consequence of practical relevance is that M-stable models of normal programs can be computed bottom-up in a process similar to the computation of the stratified model [2] of normal programs.

The modularity property of regular and M-stable models allows us to greatly extend the class of programs on which regular and M-stable models coincide. In particular, it extends the class of normal programs and the class of stratified (disjunctive) programs. This is an immediate corollary of the following theorem.

**Theorem 3** *Let $\mathcal{LP}$ be a program such that $\mathcal{LP} = \mathcal{LP}_1 \cup \mathcal{LP}_2$ and $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$, where $\mathcal{LP}_2$ is a normal program. If $\mathcal{RS}_{\mathcal{LP}_1} = \mathcal{MS}_{\mathcal{LP}_1}$, then $\mathcal{RS}_{\mathcal{LP}} = \mathcal{MS}_{\mathcal{LP}}$.*

**Proof.** By Proposition 10, each regular model $M$ of $\mathcal{LP}$ is a regular model of $\mathcal{LP}_2^{M_1}$, where $M_1$ is some regular model of $\mathcal{LP}_1$. Since $\mathcal{LP}_2^{M_1}$ is a normal program, its regular models coincide with its M-stable models by Proposition 9. By hypothesis, the regular models of $\mathcal{LP}_1$ coincide with the M-stable models of $\mathcal{LP}_1$. Thus, by Proposition 12, $M$ is an M-stable model of $\mathcal{LP}$. The proof of the converse (each M-stable model of $\mathcal{LP}$ is a regular model) is analogous. ∎

**Corollary 2** *Let $\mathcal{LP}$ be a program such that $\mathcal{LP} = \mathcal{LP}_1 \cup \mathcal{LP}_2$ and $\mathcal{LP}_2 \triangleright \mathcal{LP}_1$, where $\mathcal{LP}_1$ is (locally) stratified and $\mathcal{LP}_2$ is a normal program. Then, $\mathcal{RS}_{\mathcal{LP}} = \mathcal{MS}_{\mathcal{LP}}$.*

Another important application of the modularity property concerns model computation. Importantly, a bottom up computation of the M-stable semantics of a normal program is possible. Every normal program $\mathcal{LP}$ may be broken up into its strongly connected components [34, 28] (denoted here by $SCC(\mathcal{LP})$), which are partially ordered by the dependency relation. Informally, the strongly connected components are the smallest subsets $C$ of the rules, such that for each predicate $p$ that occurs in the head of some rule in $C$, every rule that contains a predicate mutually recursive with $p$ also belongs to $C$; they describe the finest-grained syntactical decomposition of the program $P$ into hierarchically organized modules.

More formally, mutual recursiveness is as follows. Say that a predicate $p$ depends on a predicate $q$ if they are the same, or if there is a sequence of rules $r_0, \ldots, r_k$ such that $p$ occurs in $H(r_1)$, $q$ occurs in $B(r)$, and the predicate of $H(r_i)$ occurs in $B(r_{i-1})$, for each $i = 1, \ldots, k$. Then, predicates $p$ and $q$ are mutually recursive iff $p$ depends on $q$ and $q$ depends on $p$. Notice that $SCC(\mathcal{LP})$ can be computed efficiently from $\mathcal{LP}$.

The dependency relation $\prec$ on the strongly connected components is given by the transitive closure of the natural relation $\sqsubset$ associated with them given by $C \sqsubset C'$ iff $C$ and $C'$ are different and some predicate $p$ that occurs in the head of a rule of $C$ occurs in the body of some rule of $C'$. Notice that the transitive and reflexive closure $\preceq$ of $\sqsubset$ is a partial order. The following proposition is easily verified from the definitions.

**Proposition 14** *Let $\mathcal{LP}$ be a normal program, and let $C, C' \in SCC(\mathcal{LP})$. If $C \prec C'$, then $C \triangleright C'$.*

Consequently, by applying Proposition 12, we easily obtain the following result. For each subset $\mathcal{C} \subseteq SCC(\mathcal{LP})$, denote by $\mathcal{LP}_{\prec\mathcal{C}}$ the union of all strongly connected components of $\mathcal{LP}$ on which any $C \in \mathcal{C}$ depends, and by $\mathcal{LP}_{\preceq\mathcal{C}}$ the union of $\mathcal{LP}_{\prec\mathcal{C}}$ and all strongly connected components in $\mathcal{C}$. If $\mathcal{C}$ contains a single component $C$, we write $\mathcal{LP}_{\prec C}$ and $\mathcal{LP}_{\preceq C}$ instead of $\mathcal{LP}_{\prec\{C\}}$ and $\mathcal{LP}_{\preceq\{C\}}$, respectively.

**Theorem 4** *Let $\mathcal{LP}$ be a normal program and let $\mathcal{C} \subseteq SCC(\mathcal{LP})$ whose programs are mutually independent (i.e., no $C \in \mathcal{C}$ depends on any $C' \in \mathcal{C}$). Suppose that $\max_\prec(\mathcal{C}) = \{C_1, \ldots, C_k\}$. Then,*

$$\begin{aligned}\mathcal{MS}_{\mathcal{LP}_{\preceq\mathcal{C}}} &= \cup_{M \in \mathcal{MS}_{\mathcal{LP}_{\prec\mathcal{C}}}} \mathcal{MS}_{(\cup_{i=1}^{k} C_i)^M} \\ &= \cup_{M \in \mathcal{MS}_{\mathcal{LP}_{\prec\mathcal{C}}}} \{\, M_1 \cup M_2 \cup \cdots \cup M_k \mid M_i \in \mathcal{MS}_{C_i^M}, i = 1, \ldots, k\}.\end{aligned}$$

*In particular, if* $\mathcal{C} = \{C\}$, *then* $\mathcal{MS}_{\mathcal{LP}_{\preceq C}} = \cup_{M \in \mathcal{MS}_{\mathcal{LP}_{\prec C}}} \mathcal{MS}_{C^M}$, *and if* $\mathcal{C} = SCC(\mathcal{LP})$, *then*

$$\mathcal{MS}_{\mathcal{LP}_{\preceq \mathcal{C}}} = \mathcal{MS}_{\mathcal{LP}} = \cup_{M \in \mathcal{MS}_{\mathcal{LP}-(C_1 \cup \cdots \cup C_k)}} \{ M_1 \cup M_2 \cup \cdots \cup M_k \mid M_i \in \mathcal{MS}_{C_i^M}, i = 1, \ldots, k\}.$$

**Proof.** By Propositions 12 and 14, the equivalence of the first right hand side with the left hand side of the equation is immediate. The second right hand side is equivalent to the first, since each predicate is defined by one and only one strongly connected component. (Notice also that each predicate occuring in the body of a rule in $C_i$ must be defined in $\mathcal{LP}_{\preceq C_i}$, i.e. in $C_i$ or in a SCC on which $C_i$ depends.) ∎

Theorem 4 allows a bottom-up computation of the M-stable models of a program. First split the program $\mathcal{LP}$ into its strongly connected components $SCC(\mathcal{LP})$. Then, take a linear ordering $C_1, \cdots, C_k$ of the components of $SCC(\mathcal{LP})$ compatible with the dependency relation $\prec$ of $SCC(\mathcal{LP})$ (such an ordering always exists), and compute the M-stable models of $\mathcal{LP}$ bottom-up from $C_1$ to $C_k$. In particular, the M-stable models of $C_1$ are computed first; then at the i-th step of the computation, the M-stable models of $C_1 \cup \cdots \cup C_i$ are computed by focusing on the rules of $C_i$ only, using the previously computed M-stable models as input data. Notice that, by following the partial order $\preceq$ rather than a linearization of it, part of the bottom up evaluation of the strongly connected components can also be done in parallel; thus, performance increase by parallel processing is possible.

Modularized bottom up computation is important for practical concerns, as we experienced in some projects on the implementation of deductive databases systems, namely COMPLEX [14] and the ESPRIT projects KIWI [19] and KIWIS [21, 1]. Indeed, in the context of deductive databases, where a large amount of data is assumed, it is very important to reduce the number of relations to be dealt with at a time. Theorem 4 allows to minimize this number of relations, permitting a bottom-up computation on the strongly connected components of the program, which greatly reduces the amount of data needed in main-memory during the computation.

On the other hand, designing programs in a modular fashion is natural. M-stable semantics supports this feature, while L-stable semantics does not. This suggests that from the semantical side, M-stable models are besides L-stable an appealing approximation of T-stable models.

# 7 Conclusion

In this paper, we have extended the notion of unfounded set from normal programs to disjunctive programs. By using unfounded set, we presented the notion of P-stable model for acceptable partial models for a disjunctive program. These models satisfy two principles: First, the condition of foundedness (also called justifiability [38]), and second, the principle of CWA, extended in a natural way from normal programs to disjunctive programs.

We showed that P-stable models coincide with the 3-valued stable models of Przymusinski [26]. Hence, the definition of P-stable model, which does not resort to 3-valued logic, can be seen as a characterization of 3-valued stable models in terms of classical interpretations and unfounded sets.

We next investigated refinements of P-stable models under the principle of minimal undefinedness, which has been considered in the context of normal programs [37, 29] and, for disjunctive programs, in [38]. We have proposed and analyzed maximal P-stable (M-stable) models and the more restrictive concept of least undefined P-stable (L-stable) models, which had been investigated in the course of normal programs in [32, 31].

It appeared that M-stable models are similar in spirit to the regular models in [38]. However, while regular models respect the principles of foundedness and minimal undefinedness, they do not obey to the CWA.

We have shown, however, that on a large class of programs, extending previous results, M-stable models and regular models coincide. The modularity property used for deriving this result is interesting in its own right. In particular, we obtained as a result that the M-stable models of a normal program $\mathcal{LP}$ can be computed bottom-up. This property is important for practical concerns. L-stable models lack this property. Thus, while L-stable models are stricter approximations of T-stable models (and thus preferable from the semantical side), $M$-stable models can be more amenable from the computational and methodological point of view.

The following problems are left open here. First computational aspects; in particular, expressibility and complexity of the various semantics. This issue is dealt with in a companion paper [10].

Apart from this, the design of algorithms for efficient query evaluation under M-stable and L-stable semantics is another interesting issue. Finally, further aspects on the side of knowledge representation would merit investigation, as well as fixpoint operators for computing M-stable and L-stable models (like the alternating fixpoint operator for the well-founded model [4, 35]).

**Acknowledgement** Work partially supported by the EC-US033 project "DEUS EX MACHINA: Non-Determinism in Deductive Databases" and by a MURST grant (40% share) under the project "Sistemi formali e strumenti per basi di dati evolute."

# References

[1] Ahlsen, *et al.*. The KIWIS Knowledge Base Management System. *Proc. third Int. Conf. CAiSE'91*, Trondheim, Norway, May 1991.

[2] K.R. Apt, H. Blair, A. Walker. Towards a Theory of Declarative Knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pp. 89–148. Morgan Kaufman, Washington DC, 1988.

[3] V. Baral and V.S. Subrahmanian. Stable and extension class theory for logic programs and default logic. *Journal of Automated Reasoning*, 1992, pp. 345–366.

[4] C. Baral and V.S. Subrahmanian. Dualities Between Alternative Semantics for Logic Programming and Non-Monotonic Reasoning, *Journal of Automated Reasoning*, 10, 1993, pp. 399–420.

[5] S. Ceri and G. Gottlob and L. Tanca. Logical Programming and Databases. Springer-Verlag, 1990.

[6] J. Dix. Semantics of Logic Programs: Their Intuition and Formal Properties. An Overview. In A. Fuhrmann and H. Rott (eds.) *Logic Actions and Information. Proceedings of the Konstanz Colloquium in Logic and Information*, deGruyter, Berlin, 1993.

[7] J. Dix. Classifying Semantics of Disjunctive Logic Programs. In *Proceedings ICLP-94*, 1994, pp. 798–812.

[8] J. Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12, 1979, pp. 231–272.

[9] P. Dung. Negation as Hypotheses: an abductive foundation for logic programming. *Proc. 8th Conf. on Logic Programming*, 1991, pp. 3–17.

[10] T. Eiter, N. Leone, D. Saccà. Expressive Power and Complexity of Partial Semantics for Disjunctive Deductive Databases. In preparation.

[11] T. Eiter, G. Gottlob and H. Mannila. Adding Disjunction to Datalog. *Proceedings ACM PODS-94* , May 1994, pp. 267–278.

[12] T. Eiter and G. Gottlob. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Mathematics and Artificial Intelligence*, 1995 (to appear).

[13] G. Gottlob, N. Leone, H. Veith. Second Order Logic and the Weak Exponential Hierarchies. Proc. MFCS'95, Prague, October, 1995 (to appear).

[14] S. Greco, N. Leone, P. Rullo. COMPLEX: An Object-Oriented Logic Programming System. *IEEE TKDE*, vol. 4, no. 4, August 1992.

[15] M. Fitting. A Kripke-Kleene Semantics for Logic Programs. *Journal of Logic Programming*, 2(4), pp. 295–312, 1985.

[16] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proceedings Fifth Logic Programming Symposium*, pp. 1070–1080, Cambridge Mass., 1988. MIT Press.

[17] A. Kakas, P. Mancarella. Preferred Extensions are Partial Stable Models. *Journal of Logic Programming*, 14, pp. 341-348, 1992.

[18] J.A. Fernández and J. Minker. Semantics of Disjunctive Deductive Databases. *Proceedings 4th Intl. Conference on Database Theory (ICDT-92)*, pp. 21–50, Berlin, October, 1992.

[19] KIWI Team. A System for Managing Data and Knowledge Bases. *ESPRIT Project 1117, in Proc. of the ESPRIT Technical Week*, North-Holland eds., 1988.

[20] N. Leone, P. Rullo, F. Scarcello. Unfounded Sets, Fixpoint Semantics, and Computation of Disjunctive Stable Models. *Technical Report ISI-CNR*, TR 95/1, 1995.

[21] N. Leone, P. Rullo, A. Mecchia, G. Rossi. A Deductive Environment for Dealing with Objects and Non-Monotonic Reasoning. *CRAI Tech. Report RI 94-06*, Rende, Italy, 1994.

[22] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of disjunctive logic programming.* The MIT Press, 1992.

[23] J. Minker. On Indefinite Data Bases and the Closed World Assumption. In *Proceedings of the* $6^{th}$ *Conference on Automated Deduction (CADE-82)*, pp. 292–308, 1982.

[24] J. Minker. Overview of Disjunctive Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 12, pp.1–24, 1994.

[25] T. Przymusinski. On the Declarative and Procedural Semantics of Stratified Deductive Databases. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pp. 193–216. Morgan Kaufman, Washington DC, 1988.

[26] T. Przymusinski. Stable Semantics for Disjunctive Programs. *New Generation Computing*, 9:401–424, 1991.

[27] R. Reiter. On Closed-World Data Bases. In *Logic and Data Bases*, H. Gallaire, J. Minker (eds.), Plenum Press, New York, 1978, pp. 55-76.

[28] K. Ross. Modular Stratification and Magic Sets for Datalog Programs with Negation. *Journal of the ACM*, 41(6):1216–1267, 1994.

[29] D. Saccà, C. Zaniolo. Stable Models and Nondeterminism in Logic Programs with Negation *Proc. ACM Symposium on Principles of Database Systems* 1990.

[30] D. Saccà, C. Zaniolo. Partial Models and Three-Valued Models in Logic Programs with Negation. *Proc. 1st Int. Conf. on Logic Programs and Non-monotonic Reasoning*, MIT Press, 1991, pp. 87–104.

[31] D. Saccà. The Expressive Powers of Stable Models for Bound and Unbound Queries. submitted for publication, 1994, (extended abstract in *Proc. 1993 Workshop on "Structural Complexity and Recursion-Theoretic Methods in Logic Programming.*

[32] D. Saccà. Deterministic and Non-Deterministic Stable Model Semantics for Unbound DATALOG Queries. In *Proc. ICDT 95*, Prague, January, 1995.

[33] J.S. Schlipf. The Expressive Powers of the Logic Programming Semantics. *Proc. ACM PODS Symp.*, 1990, pp. 196-204. Full paper to appear in *Journal of Computer and System Sceinces.*

[34] J.D. Ullman. *Principles of Database and Knowledge-Base Systems*, Vol. 1 and 2, Computer Science Press, Rockville, Md., 1989.

[35] A. Van Gelder. The Alternating Fixpoint of Logic Programs With Negation. In *Proceedings PODS-89*, pages 1–10, 1989.

[36] A. Van Gelder, K. Ross, and J. Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, 38(3):620–650, 1991.

[37] J. You and L.Y. Yuan. Three-valued Formalization of Logic Programming: is it needed?. *Proc. of ACM PODS Symp.*, 1990, pp. 172-182.

[38] J. You and L.Y. Yuan. A Three-valued Semantics for Deductive Databases and Logic Programs. *Journal of Computer and System Sciences*, 49, 334-361, 1994.

[39] J. You and L.Y. Yuan. On the Equivalence of Semantics for Normal Logic Programs. *Journal of Logic Programming*, pp. 211-222, 1995.